Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Desenvolupament d'un robot BB8

MEMÒRIA

Autor: Albert Carles Martin
Director: Manuel Moreno Eguilaz
Convocatòria: Setembre 2017

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

Aquest projecte té com a finalitat ser una guia detallada i completa per poder dissenyar i construir un robot tipus BB8 de la saga *Star Wars*. El projecte consta de dos documents: la memòria i els annexos.

La memòria és el document en el qual s'ha descrit tot el procés constructiu i on s'han detallat quins són i com funcionen tots els components implementats en el robot. És una guia completa de tot el procés i, d'entre altres coses, inclou imatges de les etapes de construcció, un apartat amb possibles futures millores i un detallat pressupost de tot el projecte. La memòria està dividida en tres parts: la part mecànica, la part elèctrica i la part electrònica. Cal afegir que dins de la part electrònica hi ha dos apartats dedicats a comentar els programaris desenvolupats per generar una aplicació mòbil pròpia per Android i per dotar de moviment el robot.

Els annexos complementen la memòria, aportant un recull fotogràfic extens, tots els codis font amb comentaris de l'autor dels programes o aplicacions generades i, finalment, adjunten els fulls de característiques de tots els components utilitzats.

Per altra banda, aquest projecte consta d'una important part pràctica. És per això que s'ha creat un model funcional del robot BB8, fet paral·lelament a la confecció de la memòria, i una aplicació mòbil pròpia que permet controlar a distància el robot amb qualsevol dispositiu que utilitzi el sistema operatiu *Android*.

Aquest projecte segueix l'explicació d'una guia penjada al lloc web *instructables.* Aquesta forma part d'un moviment anomenat *do it youself,* que traduït vol dir "fes-ho tu mateix". L'objectiu és que aquest projecte esdevingui una segona guia per poder desenvolupar el robot, complementant l'anterior i investigant més profundament en alguns aspectes.

Finalment, les conclusions a les quals s'arriba són que ha estat possible construir el robot a escala real amb un cost dels materials ajustat, un pressupost econòmic, i que s'ha pogut realitzar un informe detallat de tot aquest procés de construcció. Tanmateix, tot i considerar el projecte acabat, el robot no es pot donar per finalitzat, ja que té un cert marge de millora pel que fa al seu moviment i estabilitat.

Sumari

ETSEIB

1. Glossari

B4A: Basic for Android.

DIY: Do it yourself.

Paper maixé: Material fabricat amb pasta de paper barrejada amb gluten o altres substàncies enduridores, o amb fulls de paper empastats els uns sobre els altres.

BB8: Robot de la saga Star Wars. Es caracteritza per tenir el cos esfèric i el cap independent d'aquest.

Datasheet: Document que resumeix el funcionament i d'altres característiques d'un component o subsistema amb el suficient detall com per poder integrar-lo en un sistema.

Circuit integrat: Circuit electrònic implementat sobre un únic cristall de material semiconductor, el qual s'empaqueta adientment emprant encapsulats estandarditzats que disposen de terminals per a les entrades, sortides i alimentacions del circuit.

Microprocessador: És un processador o CPU materialitzat en un únic circuit integrat o xip.

Microcomputador: És un computador o ordinador la CPU del qual és un microprocessador.

Microcontrolador: És un microcomputador materialitzat en un únic circuit integrat (conté la CPU, la memòria de programa, la memòria de dades i els ports d'entrada/sortida), destinat a la implementació d'aplicacions de control.

Compilació: Procés informàtic pel qual es tradueix un programa escrit en un llenguatge de programació a un llenguatge diferent, habitualment llenguatge màquina.

IDE *(Integrated Development Environment)*: Aplicació informàtica que consisteix en un editor de codi i eines de depuració. Serveix per facilitar al programador el desenvolupament de software. En el cas d'*Arduino*, s'utilitza un entorn personalitzat i gratuït, que permet escriure, verificar, provar i compilar el software que, també el mateix aplicatiu, introdueix en el microcontrolador.

PWM *(Pulse Width Modulation)*: Tècnica en la que es modifica el cicle de treball d'un senyal periòdic per transmetre informació a través d'un canal de comunicació o per controlar la quantitat d'energia que s'envia a una carga (control velocitat motors).

Banda ISM *(Industria, Scientific and Medical)*: És un tipus de banda reservada internacionalment per un ús no comercial de radiofreqüència electromagnètica.

2. Prefaci

2.1. Origen del projecte

L'origen del projecte està en una publicació d'un fòrum [1], que promou el moviment DIY, *Do it yourself*. El que cerca aquest moviment és fer, modificar, adaptar, reparar o reutilitzar les coses amb els recursos que disposem i a través de l'autoaprenentatge. L'eslògan, *fes-t'ho tu mateix,* promou l'acció pràctica, deixant de banda la producció massiva i impersonalitzada.

Així doncs, encaminat inicialment per aquesta completa guia i gràcies, en gran part, a les ganes de fer un projecte on es puguin desenvolupar els camps de l'enginyeria que més m'agraden, vaig decidir iniciar el projecte de desenvolupament del robot BB8.

2.2. Requeriments previs

Prèviament a l'inici del treball em vaig assegurar d'estar davant d'un projecte de petites dimensions, econòmicament parlant. La intenció és obtenir un robot totalment funcional, adaptant les peces i components als estàndards que trobi pel mercat europeu, però sempre mantenint un pressupost ajustat.

Cal tenir uns coneixements mínims de programació per *Arduino* o del llenguatge del microcontrolador utilitzat. Es necessitarà també tenir un espai de treball còmode i ampli.

És un projecte molt complet: inicialment, amb una part molt manual i que requereix certes habilitats artístiques; i posteriorment, una part més conceptual i de programació, test i validació.

2.3. Estructura del projecte

Per classificar les diferents parts de la memòria s'han creat tres apartats: la mecànica, l'electrònica i l'elèctrica.

La part mecànica descriu el conjunt de components físics del robot, es a dir, el cos, el cap i la plataforma interior del robot.

L'elèctrica detalla els diferents components que gestionen o fan un ús més important d'electricitat. En són subapartats els motors i les bateries utilitzades.

Finalment, l'electrònica integra el hardware, el software i l'aplicació del dispositiu mòbil que controla el robot. El hardware del robot és la descripció del conjunt de dispositius que permeten controlar i comunicar-se amb l'exterior. El software és el programari que incorporen els dispositius per permetre que puguem entendre'ns amb aquests i programar les diverses accions que vindran donades segons les instruccions que rebi. Per acabar, l'aplicació és l'eina que permet controlar, a través del mòbil, el robot i enviar-li les instruccions.

A més a més, en els annexes, s'ha incorporat un apartat anomenat *Escandall,* que especifica el conjunt de materials que s'han utilitzat per la construcció del robot sencer.

3. Introducció

3.1. Objectius del projecte

L'objectiu que persegueix el projecte és, en primera instància, desenvolupar un prototip de robot BB8. Desenvolupar inclou tots els passos per aconseguir unir la part mecànica, elèctrica i electrònica del robot i fer que aquest es mogui seguint les directrius d'un comandament a distància.

Per altra banda, també es busca generar tot un conjunt de documentació per animar a qualsevol persona a generar el seu propi, amb les funcionalitats que cregui convenients. Aquesta documentació inclou la redacció detallada dels passos a seguir, totes les especificacions detallades dels components i, el codi de la placa *Arduino* i de l'aplicació pròpia, comentat en detall.

Finalment, es vol establir un pressupost ajustat però realista; que contempli tots els components necessaris i els materials adquirir però, a la vegada, valori les possibilitats d'utilitzar o reutilitzar tot allò que normalment tenim al nostre abast en forma de retall o residu.

3.2. Abast del projecte

La intenció és obtenir una mobilitat total per part del robot i fer que aquest s'adeqüi a la de la versió cinematogràfica, tan estèticament com en la dinàmica del moviment. Durant el procés de construcció es vol entrar en valorar diverses opcions o tecnologies per implementar una mateixa funcionalitat, intentant així fer balanços contínuament d'avantatges i inconvenients per acabar escollint la més adequada. A més a més, es vol continuar el procés que descriu la publicació del fòrum, citada en l'apartat 2.1, i incorporar o solucionar les problemàtiques que deixa pendents.

Hi ha la intenció de generar una aplicació pròpia amb una interfície còmoda i que respongui a tots els estímuls que el robot té implementats.

Inicialment, no hi ha previsió d'incorporar funcionalitats addicionals al cap que comportin l'addició d'una segona placa *Arduino* o similar per controlar-les. Aquestes, doncs, són les fronteres o límits que determinen l'abast del projecte.

4. Part mecànica

El robot BB8 està format per una esfera que constitueix el cos i que roda pel terra, i un cap amb forma de casquet esfèric, que està sempre en contacte i llisca sobre la superfície d'aquest cos. En els següents apartats es detallen els processos realitzats per tal de crear aquestes estructures.

4.1. Cos

La base del cos del robot és una pilota de platja inflable de 36 cm de diàmetre. Per començar el procés de construcció, cal assegurar-se de que la superfície està neta i intentar que estigui ben inflada per dos motius principals: el primer és que si es vol que les capes que es vagin aplicant es vagin fixant correctament, caldrà deixar-la reposar durant unes hores i això comportarà, en total, alguns dies. Durant aquest període de temps, la pilota anirà perdent una mica d'aire, però necessitem que la contracció sigui la mínima possible. El segon motiu és per evitar irregularitats en el cos que, en estar presents en una esfera que ha de rodolar per terra, podrien generar problemes durant el desplaçament o desgastar certes parts de la seva superfície amb major mesura.

La pilota es va anar recobrint de paper de diari envernissat amb una barreja de cola blanca i aigua, anomenat també *paper maixé*. El paper de diari es recomana tallar-lo a tires més o menys fines, d'uns 2 o 3 dits de gruix. La barreja d'adhesiu hauria de complir les proporcions de ¼ d'aigua per cada ¾ de cola blanca; a la pràctica i, per experiència, s'acaba obtenint la textura desitjada segons si interessa una textura més líquida (s'asseca més lent) o més espessa (més adhesiu). Va caldre fer un total de 3-4 capes completes, a tota l'esfera, per obtenir una estructura rígida. És necessari anar esperant que les capes es vagin assecant completament, ja que sinó, es corre el risc de que el paper mullat, pel seu propi pes, es vagi desprenent. Les etapes d'assecat és poden reduir temporalment amb l'ajut d'un assecador de cabells convencional, intentant no aproximar molt el flux de calor per evitar escalfar de manera perillosa la pilota de plàstic o el paper. A la Fig. 4.1 es pot veure el resultat final d'aquest procés.

Fig. 4.1. Resultat final recobriment pilota amb *paper maixé*. Font pròpia.

Un cop tot està assecat i es nota una estructura rígida en fer una mica de força de compressió a l'esfera, ja es pot procedir a recobrir el cos amb tela *canvas*. Aquesta tela és un teixit sense blanquejants forts, gruixuda i feta de cànem, lli o cotó: s'utilitza per fer tendes de campanya, veles de vaixells i com a superfície per pintar a l'oli. Es pot adquirir per metres a botigues de material artístic o cases de teles. Es van comprar 2 m^2 i va servir pel cos, el cap i en va sobrar. Per enganxar aquesta tela també es va utilitzar la barreja de cola blanca i aigua; així doncs, es va aprofitar la capacitat de la tela *canvas* d'impregnar-se de líquids fluids.

Per poder recobrir tota l'esfera amb una tela, es va haver de fer el desenvolupament en 2D d'una esfera. Com es veu a la Fig. 4.2 el desenvolupament sobre el paper són 2 arcs de circumferència que es tallen, adquirint un perfil que es podria aproximar a un romboide però amb els costats corbats, és a dir, només amb 2 vèrtexs (extrems superior i inferior). Els resultats finals del recobriment dependran de la precisió a l'hora de fer els càlculs, de la quantitat de trossos que s'utilitzaran per fer tot el desenvolupament i de la traça al retallar.

Durant el procés és possible que s'hagin de generar plecs o superposicions de la tela per adquirir la posició adequada d'aquesta.

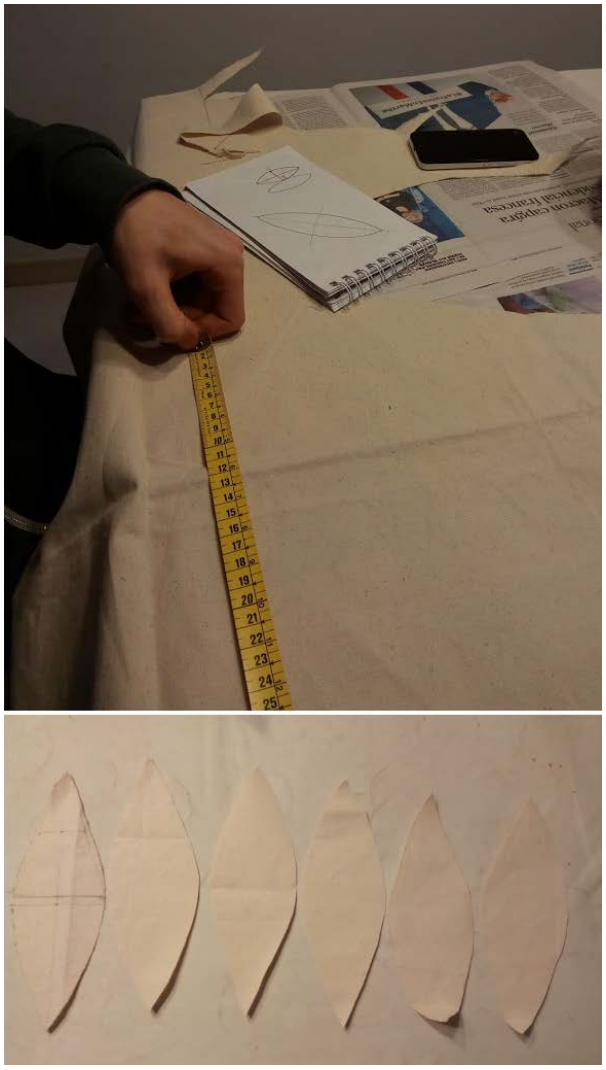

Fig. 4.2. Elaboració i perfils del desenvolupament. Font pròpia.

Com s'ha comentat amb anterioritat, aquesta tela que es va afegir té la capacitat d'absorbir, entre altres coses, la pintura, però després de l'envernissat amb cola blanca és probable que sorgeixin problemes per pintar-lo. A més a més, també s'ha dit que es van generar petits desperfectes en intentar extrapolar un desenvolupament 2D a un objecte tridimensional. Així doncs, cal utilitzar algun producte com l'*aguaplast*. En general, la intenció és buscar algun tipus de massilla que serveixi per omplir desperfectes de 2 o 3 mm de gruix.

Es recomana que s'intentin evitar les superposicions, ja que generen destacables alts i baixos a la superfície que són difícils de reduir en les següents fases. Altrament, si és necessari introduir plecs o deixar marges sense tela, de no més de 2 mm, és acceptable, ja que un cop estigui tot assecat, els plecs es poden retallar i fer desaparèixer. En el nostre cas, els petits marges o forats generats com a conseqüència de retallar un plec o per evitar posar més tela, es van poder homogeneïtzar gràcies al següent producte que es va afegir a la superfície. A la Fig. 4.3 es pot veure el resultat final de recobrir l'esfera amb tela *canvas*.

Fig. 4.3. Resultat final recobriment esfera amb tela *canvas*. Font pròpia.

Caldrà estendre bé tot el producte, per evitar posteriorment haver de llimar més i per estalviar-ne quantitat; per facilitar-ne aplicació es pot fer ús d'una espàtula de metall. Com bé s'ha dit, en aquest cas la massilla no només va servir per difuminar imperfeccions, sinó que també va ajudar a donar una última capa a tota l'esfera i fer que pintar-la fos una tasca més senzilla.

Cal comentar que es va provar amb un *aguaplast per fusta* i un *aguaplast de fibra*; aquest últim, en principi, permetia tapar fissures que tenen moviment i té una textura més plàstica. Els resultats amb el de fusta, que podem veure a la Fig. 4.4, van ser els esperats, i els requeriments per llimar la superfície després de la seva aplicació i assecat són: tenir una màquina de llimar elèctrica i utilitzar les proteccions (ulleres, guants i mascareta) per evitar la inhalació de partícules i el contacte prolongat amb aquestes, ja que va generant una petita urticària. Per altra banda, l'*aquaplast de fibra* es va utilitzar en la elaboració del cap i l'únic afegit és la necessitat de llimar la superfície amb una maquinària una mica més potent, en el aquest cas va ser una *Dremel* amb un capçat cilíndric de llima. Això fa que sigui molt més fàcil eliminar totalment el producte de la superfície i pot acabar provocant el fet d'estar contínuament donant capes per aconseguir un resultat òptim[1].

Fig. 4.4. Envàs *aguaplast* fusta i espàtula per aplicar el producte. Font pròpia.

Un cop obtinguda una superfície llisa, cal donar la primera mà de pintura. Per agilitzar aquesta primera capa genèrica es va utilitzar esprai blanc i amb dues capes ja va ser suficient. El següent pas va ser pintar les diferents sanefes i siluetes del robot. En la Fig. 4.5 s'adjunten els models que es van utilitzar. Les pintures per fer aquests detalls van ser les TITAN ACUALUX setinades, que són, bàsicament, unes pintures amb base aquosa.

[1] Les imatges corresponents a aquest procés s'han reservat per adjuntar-les en l'explicació del cap, apartat 4.2.

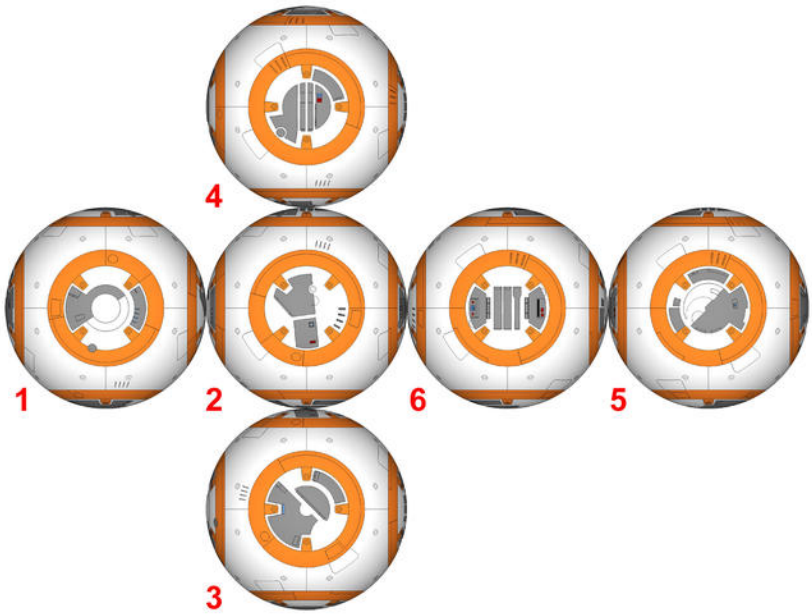

Fig. 4.5. Models sanefes externes del cos del BB8. Font: http://bb-8.blogspot.com.es/

Finalment, es va pensar en una manera de protegir l'esfera del deteriorament causat pel continu contacte amb el terra, que, en general, sempre té partícules que van erosionant la superfície. Consultant amb professionals de l'àmbit es va decidir no utilitzar cap vernís concret, per evitar augmentar el pes o impregnar l'esfera amb un producte que no fos compatible amb els materials utilitzats; simplement es va optar per donar una última capa de cola blanca, que es podrien anar repassant, generant així una protecció òptima i suficient.

Tal com es veu a la Fig. 4.6, un cop acabada la part estètica es va procedir a tallar en dues parts iguals l'esfera. Això es va haver de fer per tal d'introduir la plataforma interior, que conté els motors, les rodes i tota la part electrònica del robot. Es recomana fer-ho amb un utensili afilat i treballar amb paciència, evitant serres de molta força o moviments bruscos de fricció. Ja que la intenció en un futur era tornar-ho a ajuntar, es va ser molt curos.

Fig. 4.6. Resultat tall esfera pintada cos BB8. Font pròpia.

4.2. Cap

El procés de creació del cap del robot és similar al del cos, anteriorment descrit. Com es pot veure a la Fig. 4.7, aquest cop es va utilitzar com a base una esfera de poliestirè expandit de 20 cm de diàmetre i buida per dins. L'esfera elegida estava dividida per la meitat però per fer el cap es necessita disposar de ¾ parts de l'esfera. Així doncs va ser necessari enganxar les dues parts i, posteriorment, retallar a la mida adient. En tenir una base de polièster el cap ja disposa de certa rigidesa, i això implica que no cal recobrir-la amb *paper maixé*. Es va procedir, doncs, a generar els perfils necessaris per recobrir l'esfera amb tela *canvas*. Es va utilitzar la mateixa plantilla que amb el cos però amb les dimensions reajustades. Com es veu a la Fig. 4.8, es va fer servir cola blanca per enganxar els diferents trossos. La tècnica i les recomanacions són les mateixes que les explicades en l'apartat 4.1.

Fig. 4.7. Esfera polistirè expandit per fer el cap. Font pròpia.

Fig. 4.8. Recobriment cap amb tela *canvas*. Font pròpia.

Un cop recoberta tota l'esfera, es va aplicar una capa de massilla, que en aquest cas va ser *aquaplast de fibra*, citat anteriorment. Per llimar-ho es va utilitzar una eina de marqueteria amb un capçal cilíndric rotatiu. Aquests dos processos els podem veure a la Fig. 4.9.

Fig. 4.9. Aplicació de la massilla i procés de llima, respectivament. Font pròpia.

Fig. 4.10. Tall esfera i creació del pendent negatiu. Font pròpia.

A continuació es va tallar l'esfera a ¾ parts del diàmetre total i com s'aprecia a la Fig. 4.10, es va fer una falca de pendent negatiu respecte a l'exterior d'uns 2 cm. Això es va fer per reduir la superfície de contacte a la d'una circumferència, en comptes de mantenir el contacte amb tot el gruix de la paret externa de l'esfera. Com ja s'ha dit, l'esfera adquirida ja es va comprar buida per dins; si no fos el cas, seria necessari trobar una manera per buidar-la.

A la Fig. 4.11 es veu l'aplicació de les dues capes de pintura blanca en esprai per tota la superfície. Un cop assecades, es va procedir a afegir aquells components i/o detalls per aconseguir un millor resultat visual. En la Fig. 4.12 es veu el resultat final del cap. Com sempre, es va seguir la premissa de reutilitzar o fer servir allò que es tenia a mà. És el cas,

per exemple, dels ulls fets amb meitats de boles d'arbre de Nadal. Si les dimensions d'aquestes no fossin les adequades, es poden anar provant altres possibilitats (bales de joc, taps o simplement botons). Cal dir que es van acabar realitzant dos caps, ja que en un primer moment es va treballar amb unes mides que, més endavant, es va veure que feien desproporcionat l'aspecte del robot. Un cop detectat, es va tornar a realitzar tota la construcció per generar el cap definitiu.

Fig. 4.11. Capa de pintura al cap amb esprai. Font pròpia.

Fig. 4.12. Resultat final cap. Font pròpia.

La intenció és que el cap giri sobre el seu eix vertical. Això s'aconsegueix a través d'un joc d'imants que es mouen gràcies a l'accionament d'un servo motor que està en l'estructura del cos. L'explicació del funcionament del mecanisme i dels suports utilitzats per aguantar els imants està redactada en l'apartat 4.4.3

4.3. Plataforma i components interiors

El cos del robot té, en el seu interior, una estructura feta per suportar tots els components electrònics, motors i altres components que requereix pel seu moviment. Aquesta estructura esta formada per 2 plataformes de mides diferents i 3 columnes que les uneixen. La més gran és la que es farà sevir com a suport dels principals components electrònics i motors. La segona plataforma és més petita que l'anterior i està a pocs centímetres de la part superior de l'esfera, unida a través de les columnes a la plataforma principal. Aquesta segona plataforma té adherit el servomotor, el qual fa moure uns imants per tal de fer girar el cap, que es troba a l'exterior recolzat sobre el cos. A la Fig. 4.13 es pot veure un petit esquema realitzat amb 3D amb l'estructura comentada anteriorment. Els colors ens permeten distingir entre: plataforma gran (taronja), columnes (vermell), plataforma petita (verd). La part blanca és el suport amb els imants que gira gràcies al servomotor (negre).

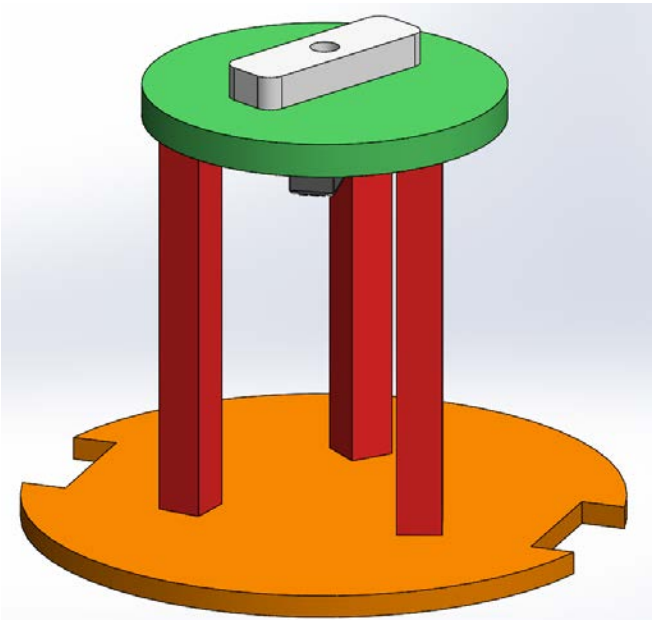

Fig. 4.13. Esquema esquelet intern. Font pròpia.

Inicialment, la plataforma interior estava feta de fusta. Es van fer proves amb altres materials i es va intentar fer una cerca per trobar-ne un que fos suficientment fàcil de treballar, sobretot de tallar, però a la vegada rígid i dur per poder ancorar correctament els motors i els components electrònics. Es van fer les primeres aproximacions amb una escuma típica de floristeria per simular sòl per les plantes, recoberta per les dues superfícies d'una planxa de fusta fina enganxada amb cola. Les planxes de fusta havien de permetre caragolar o enganxar més fàcilment altres components a la plataforma. Era un material compost lleuger però a la vegada massa gruixut. Les proves realitzades amb aquest el van descartar per suportar les forces del motor i es va considerar que, a l'anar-hi afegint components, l'estructura del material s'aniria deteriorant.

Es van buscar altres materials, tals com el poliestirè rígid o cartó ploma però, com veiem en la taula adjunta a continuació (Taula 4.1), cap aportava tants avantatges com un tros de fusta d'aglomerat.

	ESCUMA	CARTÓ PLUMA	POLIESTIRÈ	FUSTA
GRUIX	1	4	3	3
PES	2	3	4	1
RESISTÈNCIA	3	2	1	4
POSSIBILITAT CARGOLAR	3	2	1	4
FACILITAT DE MANIPULACIÓ	2	4	1	3
PREU	4	1	2	3
	15	16	12	18

Taula 4.1. Criteris per l'elecció del material. Sent 1 la pitjor puntuació i 4 la millor. Font pròpia.

Com es pot apreciar, la fusta queda primera tan en resistència com en possibilitat de cargolar. Té bastants avantatges pel que fa al gruix, facilitat de manipulació i el preu. Es va pensar que és un material que no caldria posar per capes (com el cartó ploma) i que no té cap dificultat per suportar tots els components. A més a més, la fusta és extensament utilitzada i sovint es poden tenir coneguts amb retalls o trossos que poden servir per realitzar la plataforma (preu inferior). Tot i que el pes és superior al dels altres materials, fins aquell moment era el material que aportava més confiança.

Un cop escollit el material es va tallar una circumferència de 26 cm de diàmetre. Aquesta és una mesura més petita que el diàmetre total de l'esfera del cos, ja que la plataforma havia de quedar per sota de la meitat d'aquest. A més a més, es van haver de fer unes entrades rectangulars en aquesta per permetre la instal·lació dels motors i les rodes. En la Fig. 4.2 es detallen les mides i les formes anteriorment descrites.

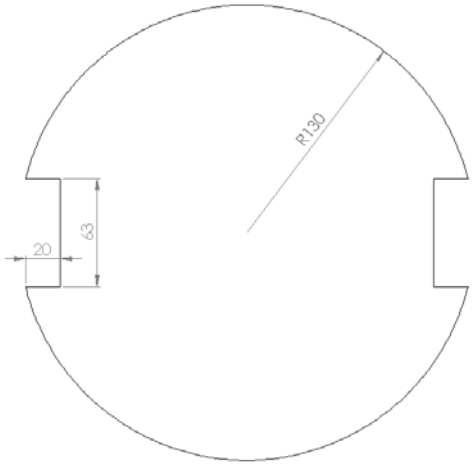

Fig. 4.14. Plànol plataforma gran. Font pròpia.

Així mateix, la distribució dels components a la plataforma també es va intentar que fos la més adequada per equilibrar el pes i evitar que la inèrcia generi problemes d'estabilitat del robot. A la Fig. 4.15, veiem tota l'estructura de fusta incloent les dos plataformes però sense ancorar els components.

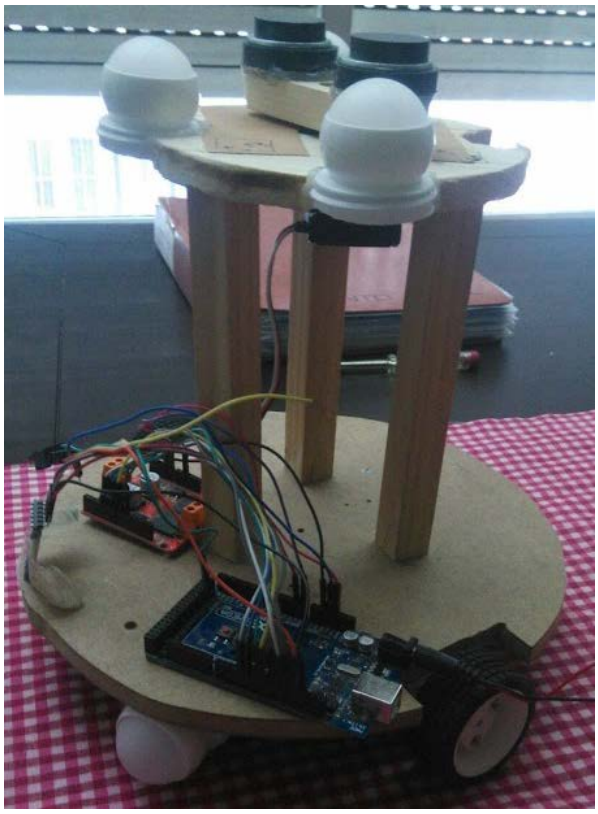

Fig. 4.15. Estructura de fusta. Font pròpia.

Fig. 4.16. Estructura de plàstic. Font pròpia.

Un cop es van distribuir tots els components i donat que, simultàniament, s'estaven realitzant models en 3D dels eixos de la roda (4.4.2), dels suports dels imants (4.4.3) i de la finestra externa (4.6), es va aprofitar per repensar altre cop el material de les plataformes i intentar modelar també tota l'estructura. Fent-ho així ja es tenia clara quina havia de ser la millor distribució dels components i es podia fer servir aquesta informació per generar un component més optimitzat. Aquesta elecció també va venir motivada en veure les propietats del material plàstic que s'utilitza per a la impressió en 3D. A l'anterior figura (Fig. 4.16) es pot veure tot l'esquelet intern al cos fet amb material plàstic.

Actualitzant la taula presentada anteriorment, obtenim els següents resultats (Taula 4.2)

	FUSTA	IMPRESSIÓ 3D
GRUIX	1	2
PES	1	2
RESISTÈNCIA	2	1
POSSIBILITAT CARGOLAR	2	1
FACILITAT DE MANIPULACIÓ	1	2
PREU	1	2
	8	10

Taula 4.2. Annex taula de criteris per l'elecció del material, sent 1 la pitjor puntuació i 2 la millor. Font pròpia.

El material utilitzat en les impressions 3D és l'àcid polilàctic (*PLA*), que és un polímer termoplàstic rígid i biodegradable. Un cop es carrega un model concret a la impressora, aquesta programa els moviments dels capçals per imprimir-la. Els capçals escalfen el material fins la seva temperatura de fusió i el van distribuint per aconseguir la forma desitjada. En ser un material plàstic té una densitat menor i, per tant, el seu pes disminuiria respecte el de la fusta. A més a més, la peça serà estrictament idèntica a com s'ha dissenyat (amb toleràncies) i això permet establir la distribució dels components fent ja els forats corresponents. També es poden generar caixes protectores per els components petits com el mòdul *Bluetooth*, per components que són complicats de subjectar com les bateries o per fer una cavitat per posar l'interruptor.

En ser una plataforma tan gran es va haver de tallar en dos parts i, per garantir una bona unió entre aquestes, es va dissenyar un tall esglaonat. Això es va fer perquè, un cop acabada la impressió, hi hagués suficient superfície llisa per poder posar-hi cola i obtenir una bona subjecció. A la Fig. 4.17 es fa un petit recull d'aquests ginys generats i implementats.

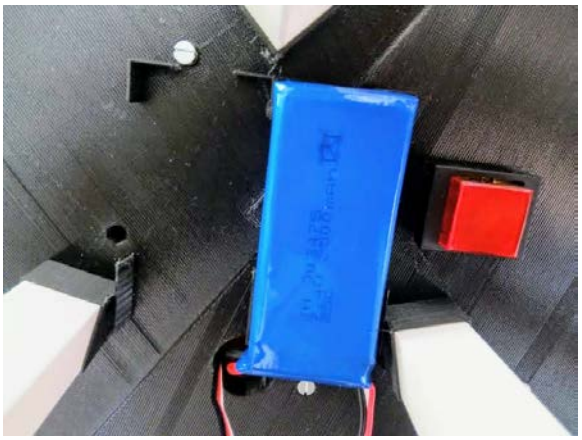

Fig. 4.17. Disseny plataforma des d'una perspectiva d'ocell. Font pròpia.

4.4. Altres components

4.4.1. Components per disminuir la fricció: els roll-on

Uns components importants que es van afegir a les dues plataformes van ser els roll-ons que permeten eliminar el contacte entre la vora de les plataformes i l'interior de l'esfera, facilitant-ne el lliure moviment i disminuint així la fricció.

A la plataforma principal, s'hi van posar 4 roll-ons en un angle de separació de 90º i, separats, inicialment, 45º dels motors. A la plataforma superior només en van ser necessaris 3 en posició vertical. Els roll-ons es poden reciclar de desodorants acabats o adquirir-ne a qualsevol supermercat. Aquests s'han de tallar, ja que només és necessària la part de la bola. Per enganxar-los a la plataforma es va fer servir una pistola de cola calenta. En la Fig. 4.18 s'hi pot veure la distribució d'aquests components en la plataforma superior.

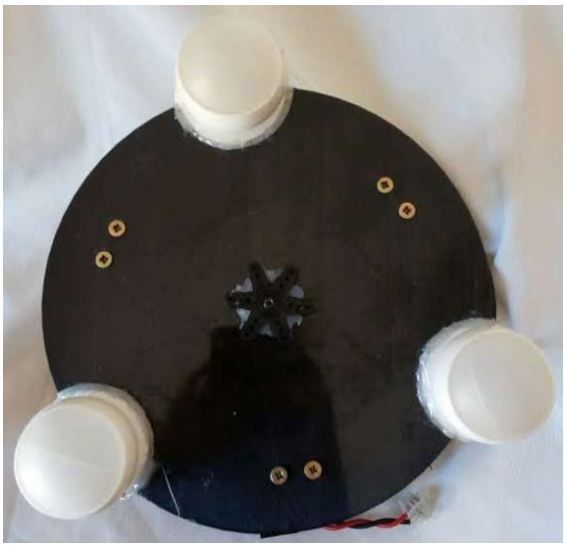

Fig. 4.18. Distribució dels roll-on en la plataforma superior. Font pròpia.

4.4.2. Unió del eix del motor amb la roda

En l'apartat 5.1 s'expliquen els motors utilitzats per fer moure el robot. Aquests motors tenen uns eixos que han de fer girar unes rodes. Aquestes rodes, que estan en contacte constant amb l'interior de l'esfera, fan girar tot el cos del robot i que aquest avanci. Les rodes es van adquirir juntament amb els motors, però en el moment de la compra no es va revisar si els adaptadors pels eixos que incorporaven a part eren de les dimensions correctes. En veure que no, es va començar a cercar solucions per aprofitar les rodes. Finalment, es va optar per fer un disseny 3D basat en les mesures reals de l'eix, però prenent de model les unions eix-roda que incorporaven les pròpies rodes.

Cal dir que els eixos de les rodes són uns components que van ajustats i els quals, degut a les toleràncies de la pròpia impressió, van ser impresos repetidament per tal d'intentar millorar la seva estructura i fermesa. Es creu que degut al ampli catàleg d'impressores que hi ha a la universitat i a causa de les diverses toleràncies que tenen cadascuna d'aquestes, no

es va poder aconseguir imprimir un component amb les mesures exactes i, finalment, es van haver d'adaptar manualment a les dimensions requerides.

El material utilitzat per a la impressió ja s'ha explicat en l'apartat anterior i el servei d'impressió utilitzat va ser el de REP RAP de la universitat.

A la figura adjunta es mostra el plànol d'un component eix-roda així com el resultat final obtingut (Fig. 4.19).

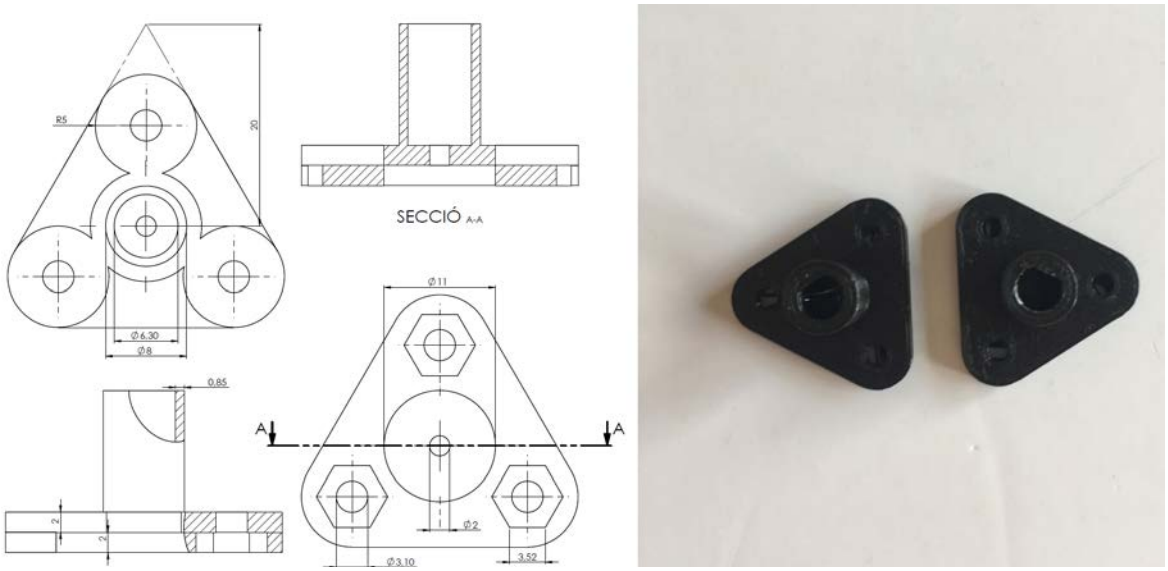

Fig. 4.19. Plànol i impressió definitiva del component eix-roda. Font pròpia.

4.4.3. Gir del cap: mecanisme i suports pels imants.

En explicar les plataformes interiors del robot (4.3) s'ha esmentat que el cap estava recolzat sobre la superfície externa de l'esfera. Això és possible gràcies a un mecanisme d'imants que a través del magnetisme entre ells permeten, sense cap contacte directe, mantenir el cap sobre el cos fins i tot quan aquest es mou. El joc d'imants a més a més també permet, a través d'un servo motor, girar el cap sobre el seu eix vertical ens els 360º. Per tal de fer-ho possible, la segona plataforma (la més petita) té adherida un servomotor que fa girar una pastilla rectangular amb 2 imants enganxats en els extrems, tal com veiem a la Fig. 4.20.

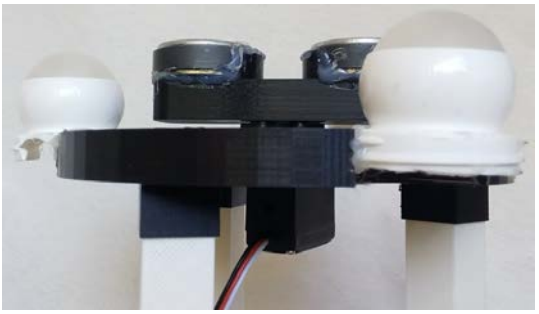

Fig. 4.20. Mecanisme d'imants pel gir del cap. Font pròpia

Els imants es van reciclar d'amplificadors de música obsolets o que ja no funcionaven. Aquests imants interactuen amb uns altres imants de l'exterior que, com veiem a la Fig. 4.21, es troben adherits al cap a través d'un suport fet a mida i imprès amb tecnologia 3D.

Fig. 4.21. Suport imants cap. Font pròpia

Els imants externs són més petits i de neodimi. Per col·locar-los només cal tenir els cos muntat i deixar-los anar sobre la superfície esfèrica esperant que interactuïn magnèticament amb els imants interns.

A la següent figura podem veure l'aspecte del conjunt muntat (Fig. 4.22).

Fig. 4.22. Aspecte final del mecanisme. Font pròpia

4.5. Accés a l'interior un cop assemblat

Des dels inicis, una de les principals problemàtiques del robot va ser com accedir a l'interior d'un cos esfèric que ha d'estar tancat i que no pot tenir cap tipus d'imperfecció interna o externa per evitar problemes en el moviment. La solució immediata va ser pensar que el cos es tancaria i que, si hi havia problemes o avaries, ja es tornaria a obrir, però llavors es va haver de reflexionar sobre si interessava un tancament definitiu o un temporal. Definim tancament definitiu com el procés de reunir les 2 meitats esfèriques a través de paper de diari amb cola i massilla, repintar i fer desaparèixer qualsevol evidencia de separació. Per altra banda, definim tancament temporal com el procés de tancament que és reversible i estructuralment estable.

Es van pensar diverses maneres de fer tancaments temporals, però totes elles incloïen components que sobresortien de la superfície llisa i/o perjudicaven el moviment del robot. Una de les idees més realistes i amb possibilitats de dur a terme era la de dissenyar i imprimir amb 3D tota la superfície del cos, és a dir, tota l'esfera. Tanmateix, igual que en el cas d'altres components més petits com els eixos o les plataformes sí que es va tirar endavant aquesta tècnica, es va pensar que en aquest cas ja es tenia un cos fet i pintat i no es disposava de més temps per fer les diverses proves.

Descartada l'opció del tancament temporal, es va fer un anàlisi de les possibilitats d'accés amb un tancament definitiu. La opció que va convèncer més va ser la de generar una finestra, aprofitant les sanefes rodones que hi ha pintades a l'esfera, que es pogués descargolar o obrir per tal de poder-hi posar les mans i manipular el que fos necessari. Tasques com la recarrega de les bateries, la comprovació del hardware o l'actualització del software es tornaven viables. S'entenia però, que qualsevol canvi important passaria obligatòriament per tornar a obrir completament el robot.

Es van fer un total de cinc proves abans d'obtenir la finestra definitiva. Aquestes impressions van comprovar les toleràncies de la impressió i els tipus de tancament que podia tenir la finestra. S'ha de tenir en compte que el gruix del cos és de menys de 5mm, això implicava que en aquests pocs mil·límetres hi havia d'haver una rosca o un passador suficientment gruixut per ser imprès i no trencar-se. Es van definir dos components separats: un marc que anava enganxat a l'esfera i la pròpia finestra, que s'extreia per deixar el forat d'accés. El procés de disseny va ser el següent:

- Es va fer una primera aproximació amb un tancament entre marc i finestra fet amb una rosca helicoïdal de 360º. Aquesta proposta va resultar tenir un problema gran amb la fricció entre els components, ja que en tenir tanta superfície en contacte era impossible de obrir o tancar fàcilment.

- A continuació es va intentar retallar el recorregut de la rosca. Es va optar per fer 2 rosques helicoïdals simètriques de 90º de recorregut. Igualment es trobava un problema en el marc, ja que el gruix era massa petit i hi havia zones que no s'imprimien correctament.

- Repensant el disseny es va trobar una idea que solucionava els problemes de fricció i mida d'impressió. Aquesta era fer passadors distribuïts uniformement per la circumferència exterior de la finestra. Els passadors tenien un recorregut d'uns 5º i constaven d'una part més gruixuda però curta i d'una més llarga però fina que entrava dins d'una incisió que tenia feta el marc.

- Després de dues proves realitzades amb aquest tancament es va decidir posar 3 passadors separats 120º i aplicar-hi unes toleràncies de seguretat per assegurar la seva unió.

A la Fig. 4.23 es pot veure una mostra de les proves detallades anteriorment. L'última correspon al disseny final amb les toleràncies correctes però sense una impressió completa.

Fig. 4.23. Proves component marc i finestra amb ordre cronològic. Font pròpia

A continuació es presenta la Fig. 4.24, on podem veure la implementació final sobre el cos.

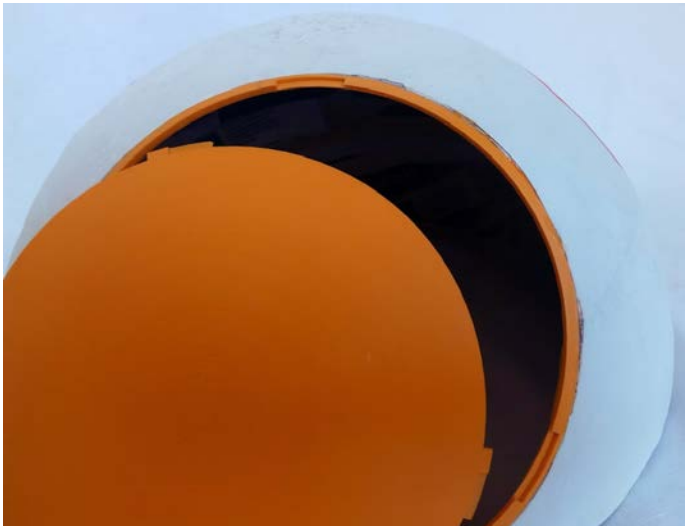

Fig. 4.24. Aspecte final del mecanisme. Font pròpia

4.6. Tancament del cos

El tancament del cos va ser la part més delicada del projecte. A l'hora de tallar les dues meitats de l'esfera del cos es va fer una marca comuna a les dues parts per, en un futur, saber per on s'havien de fer coincidir ambdues. Hi havia un cert neguit per veure si, passat un temps i sumant el pes de l'estructura interior, a les dues meitats els hi costaria més coincidir, ocasionant problemes en aquest aspecte. La intenció inicial era fer un tancament definitiu posant-hi paper amb cola, *aguaplast* i tela per tancar la fissura feta pel tall. Consultant aquesta opció amb agents externs, es va veure que el robot, tot i tenir una finestra suficientment gran per manipular els components interiors, s'havia de poder obrir totalment per qualsevol emergència major, i no podia ser que cada cop s'hagués de tancar d'aquella manera. A més a més, es va valorar que seria molt complicat assegurar-se que la fissura quedava totalment rígida, ja que amb el procés anteriorment descrit, si no es volia augmentar molt el gruix, es farien poques capes i acabaria sent una solució més estètica que no pas mecànica. Es va optar per unir les dues parts amb cinta americana, un tipus de cinta molt resistent que té una estructura interna similar a la xarxa que forma un teixit. Aquesta és suficientment resistent, econòmica i fàcil de treure i posar en cas d'emergència; i, en tot cas més endavant, es podria valorar el fet d'amargar-la amb *aguaplast* i pintura.

Abans de començar a tancar l'esfera, es va detectar un problema que impossibilitava seguir amb el procés. La mida de les columnes era superior a la necessària i, per uns mil·límetres, deformava i impedia la unió de les dues meitats. Les columnes fetes amb impressió 3D no es podien tallar, ja que estaven impreses, per estalviar material, amb una estructura semblant a les dels ruscs d'abelles. Si aquestes es tallaven, es posava en perill la seva rigidesa i els caragols de les plataformes no s'haurien pogut caragolar. En aquell moment es va valorar que seria impossible dissenyar qualsevol columna de mides exactes, ja que l'esfera estava feta a mà i hi havia moltes possibilitats de trobar imperfeccions o no tenir les mesures al mil·límetre. Es van utilitzar, doncs, les antigues columnes de fusta, que igualment tenien una mida que no coincidia amb el que necessitàvem. Sent conscients d'aquest problema es va valorar afegir un material, a una de les puntes de cada columna, capaç de donar una mica de joc als caragols i permetent així adaptar les columnes, un cop caragolades, segons la necessitat de cadascuna d'elles i del conjunt de l'esquelet. La Fig. 4.25 mostra aquestes columnes.

Fig. 4.25. Columnes finals. Font pròpia

Aplicant aquest recurs es va obtenir un esquelet intern que no impedia el tancament de l'esfera i es va procedir amb aquest. Com ja s'ha comentat es va utilitzar cinta americana: aquesta es va tallar en petites tires que s'adaptaven bé al tros d'esfera que havien de cobrir. S'intentava que quedessin juntes i centrades respecte el tall. L'aplicació de la cinta adhesiva es va fer tan per la superfície exterior com per l'interior, treballant en paral·lel. Per assegurar una major rigidesa es van anar dipositant punts o zones amb cianoacrilat[2]; un adhesiu d'enduriment ràpid. Amb paciència el tancament va ser tot un èxit i va quedar suficientment fort per poder rodolar i suportar el moviment del conjunt.

En cas d'emergència l'obertura serà fàcil de realitzar i el posterior tancament també. Cal dir que la cinta adhesiva també es fa servir per donar un suport extra a la finestra, ja que es va veure que els passadors eren molt fins i es podien trencar en passar l'estructura interior per sobre. A la Fig. 4.26 es mostra l'aspecte actual del robot sense dissimular, de moment, la cinta americana.

Fig. 4.26. Aspecte extern actual del robot. Font pròpia

[2] La marca comercial utilitzada va ser *Super-Glue.*

5. Part elèctrica

5.1. Motors *POLOLU 1102*

Un motor de corrent continu és un sistema electromecànic que permet transformar energia elèctrica en energia mecànica mitjançant interaccions magnètiques. Els motors utilitzats són els mateixos que s'indicaven en la guia [1], que amb anterioritat ja s'han introduït. Això es va fer per assegurar que aquests tenien la suficient força per a moure còmodament el robot. Les característiques i dimensions dels motors es presenten a continuació.

Són motors de corrent continu de tensió nominal 12 Volts amb possibilitat d'inversió de voltatge, la qual cosa permet canviar el sentit de gir. Les característiques tècniques principals són les 500 rpm i els 0,3 A que consumeix sense carrega. Les dimensions són 37 mm de diàmetre i 52 mm de longitud (sense eix). L'eix té forma de D.

Tal com veiem a la Fig. 5.1, són uns motors d'una mida raonable i la potencia és suficient per moure tota l'estructura.

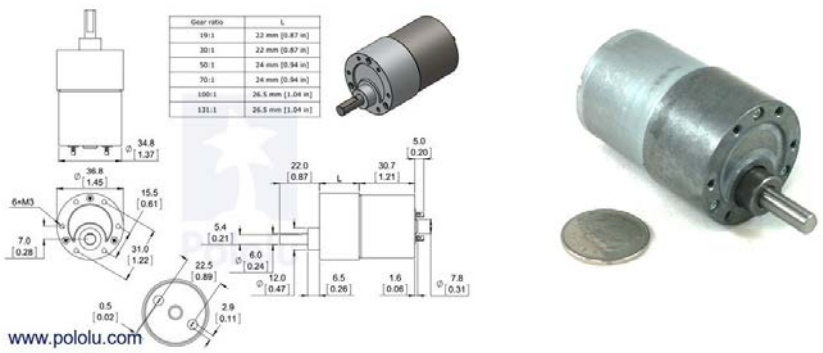

Fig. 5.1. Plànols i imatge real motors Pololu. Font : www.pololu.com

Els motors es van adquirir en un botiga en línia, juntament amb les rodes i uns suports per aguantar-los. Aquests últims, tal com veiem a la Fig. 5.2, es van cargolar a la plataforma, donant més estabilitat a tota l'estructura. La documentació de tots aquests components s'adjunta a l'annex.

Fig. 5.2. Suport motor. Font pròpia

5.2. Servomotor *FUTABA S3003de les*

Un servomotor és un sistema electromecànic que permet controlar la posició del seu l'eix. És un motor especial dissenyat per moure's una determinada quantitat de graus, permetent mantenir-se en una posició fixa si fos necessari [2]. S'ha fet servir un per moure el mecanisme d'imants que fan girar el cap (4.4.3).

El servomotor es va cargolar a la plataforma superior, que és més petita que la principal i que s'uneix amb aquesta a través d'unes columnes, tal com s'explicava en l'apartat 4.3. Aquesta plataforma també està feta amb 3D i té un forat dissenyat per col·locar-hi el component. El servomotor rep les instruccions de l'*Arduino*. La connexió entre aquests dos és a través de 3 cables: un vermell pel voltatge requerit de 5 V, un negre que es connecta al terra (*ground* o *GND)* i l'últim blanc que rep els impulsos PWM que fan girar l'eix.

Per aquesta versió del robot no era necessari controlar els graus girats pel servomotor, però es reserva aquesta característica per possibles futures implementacions. A més a més, no es requeria de més potencia i, tot i ser uns motors senzills, funcionaven pel que es necessitava. Tanmateix, tot i que el model original no gira els 360º, el nostre model està manipulat per aconseguir el gir complet. En la Fig. 5.3 veiem una imatge del servomotor elegit.

Fig. 5.3. Servomotor Futaba S3003.
Font: REES52

5.3. Bateries

5.3.1. Implementació inicial

Inicialment, per fer les primeres proves, es va utilitzar una caixa de 8 piles AA d'1,5 V. Això feia un total de 12 Volts que alimentaven els 2 motors i la placa *Arduino*, juntament amb el dispositiu *(Shield),* que controla la potència transferida als motors. La informació sobre aquests components electrònics s'ampliarà en els següents apartats.

Amb aquesta implementació es va suposar un voltatge màxim requerit de 12V, però no es van fer els càlculs de les diferents necessitats d'amperatges ni la possibilitat de construir un

mesurador de càrrega romanent. Senzillament, es va mirar un corrent que no fes malbé cap components i es va utilitzar una tecnologia que permetia fer moltes proves i tenir recanvis; les piles. A la pràctica és un sistema poc òptim. A més a més, suposen una càrrega extra a la plataforma i era necessari trobar altres solucions.

5.3.2. Solució definitiva

Després de veure els problemes i la possibilitat de millora dels sistema anterior, es va plantejar fer servir bateries tipus LiPo per alimentar totes les parts electròniques i els motors del robot. Aquestes bateries estan fetes de polímers de ions de liti, que basen el seu funcionament en l'intercanvi d'electrons entre el material amb l'elèctrode negatiu i el material amb l'elèctrode positiu mitjançant un medi conductor [3][4]. Per evitar el contacte directe entre els elèctrodes s'utilitza un material amb porus microscòpics que només deixa passar els ions. La principal diferencia amb la resta de bateries de liti és el medi conductor que aquestes utilitzen: en el cas de les bateries tipus LiPo s'utilitza com a electròlit (medi conductor) un material polimèric en estat sòlid (SPE o *solid polymer electrolyte*), per altra banda les bateries de liti comunes utilitzen com a electròlit una solució líquida de salt de liti com el compost $LiPF_6$.

L'avantatge d'aquest tipus de bateries són:

- Són bateries lleugeres que poden adoptar qualsevol mida i forma.
- Tenen gran capacitat respecte la seva mida.
- Tenen una taxa de descarrega alta, així que poden nodrir sistemes elèctrics exigents.

Igualment, aquestes bateries també tenen inconvenients, com per exemple problemes de seguretat al contenir un electròlit volàtil i, sobretot, el fet de requerir un manteniment exhaustiu per allargar la seva esperança de vida.

Per fer front a tots aquests desavantatges, les bateries es van instal·lar a la cara oposada d'on hi ha tota l'electrònica. És una zona més oberta i, en cas d'escalfament, no afectaria a altres components. A més a més, la recarrega de les bateries es durà a terme amb un dispositiu extern que subministra específicament la intensitat i el voltatge requerits per les bateries per no desequilibrar els seus valors de voltatge màxim i mínim.

Després d'analitzar les necessitats especifiques del sistema electrònic complet a través dels aparells de mesura del laboratori, es va establir que el consum màxim estava al voltant dels 2 A i el voltatge requerit al voltant dels 12 V. Finalment, les bateries adquirides van ser 2 bateries idèntiques tipus LiPo de 7,4 V i 2500 mAh. Aquestes es van connectar en sèrie per

tal d'obtenir un voltatge d'entrada màxim de 14,8 V i una capacitat de 2500 mAh. Veiem una d'aquestes bateries a la Fig. 5.4.

Després de consultar la *datasheet* [5] de la placa *Arduino*, dels motors POLOLU 1102 i de la Shield VNH2SP30 es va comprovar que aquest voltatge no posava en risc el correcte funcionament d'aquests component, i que tot i no ser el voltatge nominal requerit, entrava dins dels marges de seguretat establerts en les fulles de característiques. Aquestes fulles de característiques s'adjunten en l'annex.

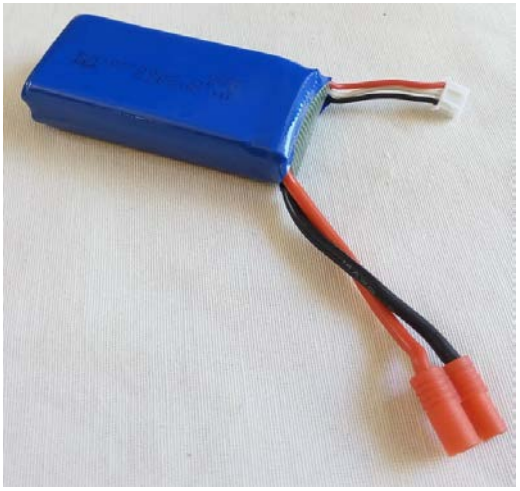

Fig. 5.4. Bateria LiPo implementada. Font pròpia.

Els 2500 mAh que subministren les bateries permetrien el funcionament continu del robot durant aproximadament una hora i quart. Cal afegir que els 2 A es van establir com un valor a l'alça, així que s'espera un temps inclús superior.

5.3.2.1. Recàrrega

En ser un sistema d'alimentació propi i recarregable, s'ha de tenir en compte com fer la recarrega per tal que aquesta es pugui repetir el màxim de cicles possibles i allargar així l'esperança de vida de la bateria.

Com s'ha comentat en l'apartat 5.3.2, la recarrega es realitza amb un component extern adquirit simultàniament amb les bateries i que està específicament fabricat per recarregar bateries tipus LiPo amb un voltatge pròxim als 10 V, tal i com es pot apreciar a la Fig. 5.5. Aquest dispositiu es connecta, a traves d'un transformador, a la xarxa elèctrica. El dispositiu té dues entrades: una per connectar-se amb el transformador anteriorment esmentat i l'altre que permet la connexió d'un component rectangular de 3 pins com el que té la bateria. Tal i com hem visualitzat a la Fig. 5.4, la bateria té 2 sortides: una amb la connexió rectangular, que serveix per la seva recarrega, i l'altre formada per 2 cables (vermell i negre), que subministren l'energia emmagatzemada a on es requereixi.

Fig. 5.5. Carregador de les bateries LiPo escollides. Font pròpia.

6. Part electrònica

6.1. Hardware

6.1.1. Components utilitzats

6.1.1.1. Arduino MEGA 2560

Arduino és una empresa de hardware lliure. Aquesta dissenya i manufactura plaques simples formades per un circuit imprès on hi ha un microcontrolador. A més a més, ofereixen un entorn de desenvolupament (IDE) *open-source*, capaç de compilar el software desenvolupat per un agent extern; cal dir que, actualment, *Arduino* ofereix una eina online anomenada *Arduino Web Editor* [6].

Arduino té com objectius apropar i facilitar l'ús de l'electrònica i programació en projectes multidisciplinaris, però que no requereixin d'uns dispositius molt potents. El llenguatge que aquest entorn entén està basat en C; suporta totes les funciones estàndards de C i algunes de C++. A la Fig. 6.1 es mostra un codi d'exemple per fer parpellejar cada segon el LED 13 de la nostra placa; aquest està escrit en l'entorn IDE per *Arduino*.

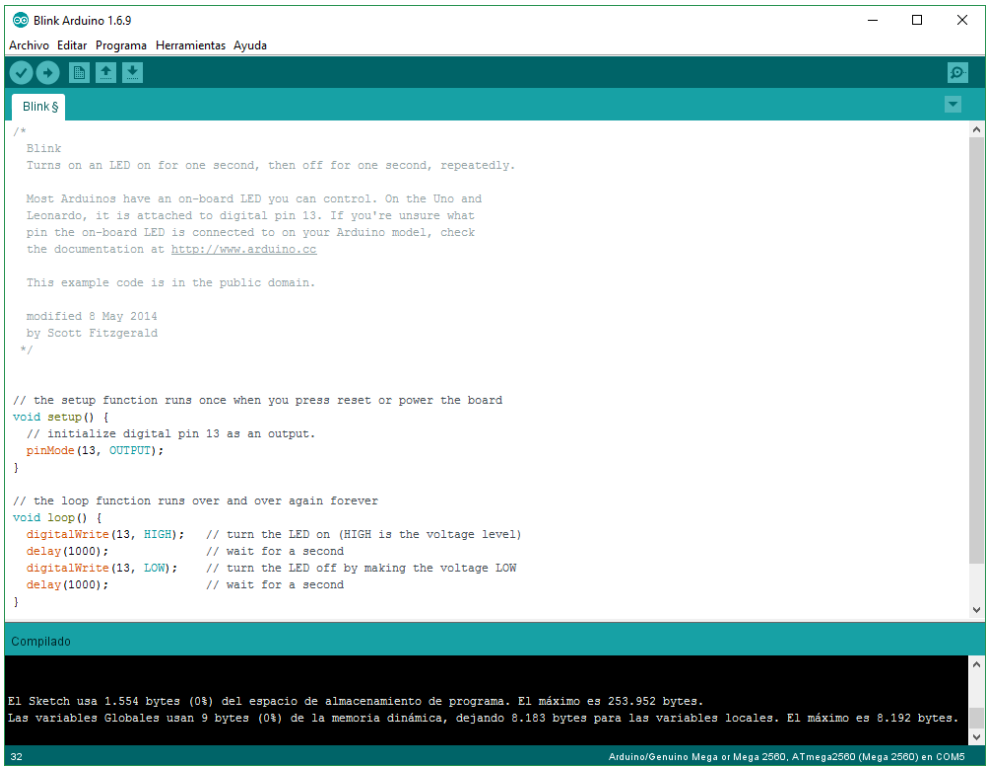

Fig. 6.1. Exemple codi *Arduino* en entorn IDE *Arduino* 1.6.9. Font pròpia.

Hi ha models i dimensions de plaques per totes les necessitats. Aquestes disposen de més o menys ports d'entrada/sortida que permeten interaccionar amb l'exterior. Normalment, com a mínim, totes elles disposen d'un port USB que permet la comunicació amb l'IDE i una entrada de voltatge - *ground*.

L'elecció per aquest robot va ser una placa *Arduino* MEGA 2560, basada en un microcontrolador ATmega2560, que podríem dir que és una de les més completes. A la Fig. 6.2 hi ha uns vista esquematitzada d'aquesta i, a continuació, s'expliquen el conjunt d'entrades i sortides més importants que la placa té.

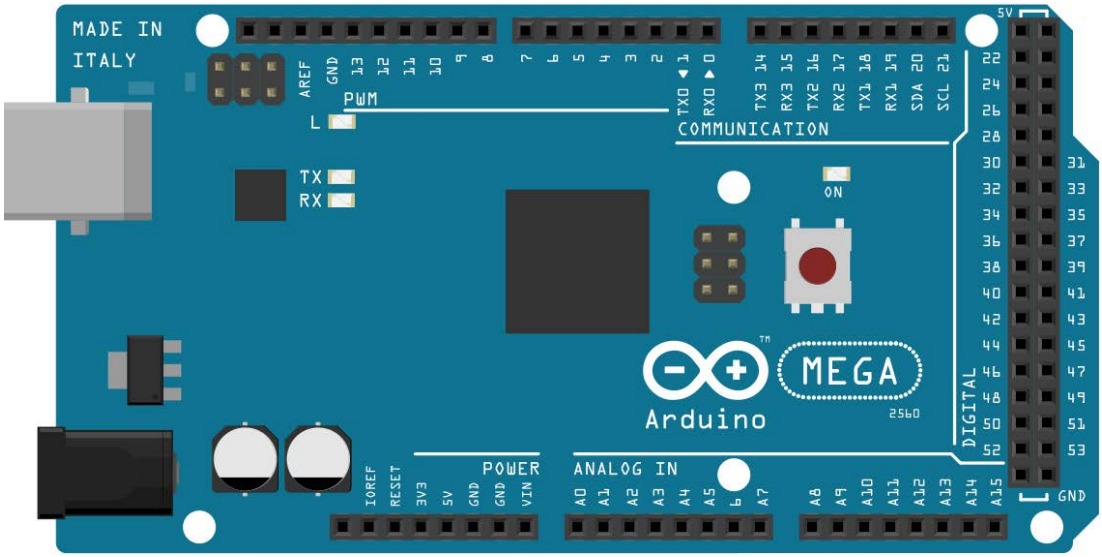

Fig. 6.2. Esquema realista de la placa *Arduino* MEGA 2560. Font: **Patagonia**tec

Els pins *power* són els que proporcionant voltatge a l'exterior. N'hi ha de 5 Volts i de 3,3 Volts, l'elecció depèn del component a connectar. També trobem els pins de GND *(ground)* que serien el negatiu o terra.

Els pins *analog in* són els que proporcionen una entrada o sortida analògica. Una entrada/sortida analògic es caracteritza per tenir infinits valors diferents. Aquests pins es marquen amb una A i un número del 0 al 15.

Els pins *digital* són els que proporcionen una entrada/sortida digital. Una entrada/sortida digital es caracteritza per tenir un nombre finit de valors, en aquest cas binari.

Els pins *communication*, anomenats també pins sèrie, són un tipus de pins digitals que s'utilitzen per comunicar-se amb l'exterior, fan ús de la tecnologia lògica transistor a transistor o TTL [7]. El pin TX envia dades i el pin RX rep dades. En la placa Mega2560 tenim 3 parells de *serial pins (comunicació per UART).*

Els pins *PWM* són un tipus de pins digitals que ofereixen la possibilitat d'obtenir valors analògics a partir de senyals digitals periòdics modulats en amplada. S'utilitzen per controlar la velocitat dels motors o per enviar informació.

Tenim un pin especial: el 13. Aquest pin té incorporat un LED amb una resistència associada, que ens permet fer comprovacions rapides o provar comportaments sense afegir més components que el sistema extern que estem testejant. Un exemple d'ús podria ser provar un mòdul detector d'obstacles (LED on/off) o un de proximitat d'objectes (intensitat LED).

A més a més, l'*Arduino Mega2560* inclou un port USB i una entrada de 2,1 mm de tipus *center-positive* que permet alimentar l'*Arduino*. El voltatge d'alimentació recomanat és de 7-12 V, que poden venir donats pel port USB, pel connector de *center-positive* o a través dels pins Vin i GND que es troben amb els pins *power*. En la Fig. 6.3 es pot veure la placa real amb els connectors anteriorment esmentats.

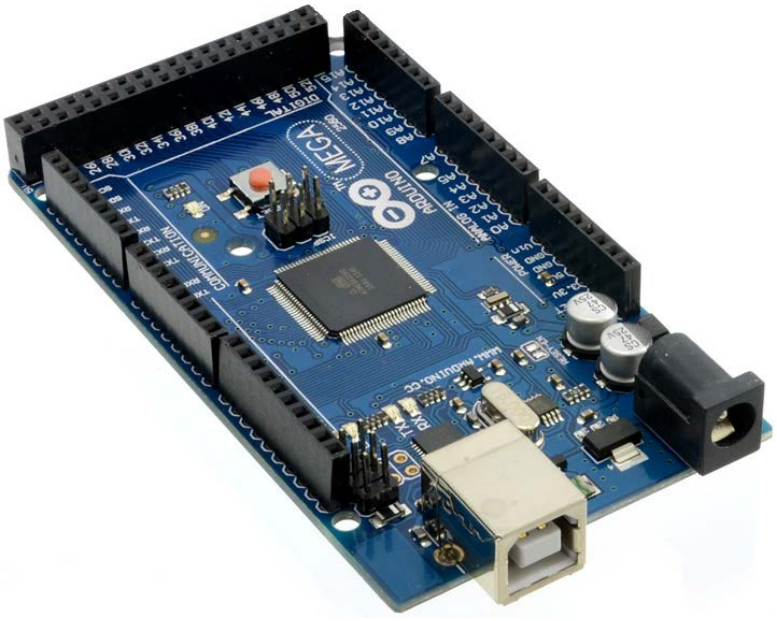

Fig. 6.3. *Arduino Mega2560* real amb connectors d'alimentació. Font: **LAMPA**TRONICS

6.1.1.2. Mòdul Bluetooth HC-05

El mòdul HC-05 és un component extern utilitzat per vincular-se amb el telèfon mòbil i rebre les instruccions d'aquest mitjançant *Bluetooth*.

El *Bluetooth* és un tipus de xarxa sense fils d'àrea personal que permet la transmissió de veu i dades a través de radiofreqüència, en la banda ISM de 2,4 GHz.

Aquest component té un model similar al mercat molt semblant anomenat HC-06. La

diferencia entre aquests dos models és la capacitat que té un, el HC-05, de poder-se posar en mode esclau o màster. Amb aquesta funcionalitat, el component pot rebre instruccions d'un dispositiu (*slave* o esclau) o enviar-ne a un altre dispositiu (*master* o màster). Tanmateix, en el nostre cas el mode emprat és el d'esclau, així doncs, seria independent el model. Per previsió de futures millores i donada la poca variació de preu es va elegir el model HC-05.

A la Fig. 6.4 veiem una imatge d'aquest component:

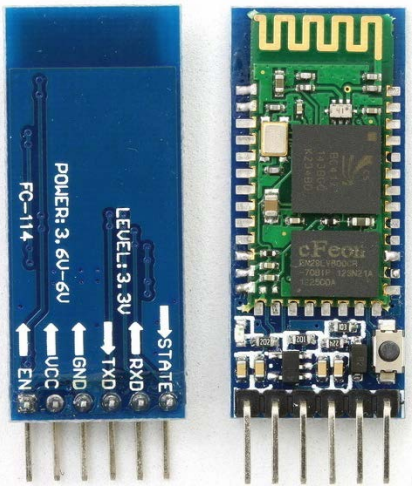

Fig. 6.4. Mòdul HC-05. Font: Blog MartynCurrey

Com es pot apreciar, el mòdul té 6 pins, els quals detallarem a continuació:

- **STATE**: Permet veure l'estat del mòdul. És un pin de sortida i dispensable pel seu funcionament.
- **TX**: Permet enviar informació del component a la placa *Arduino*.
- **RX**: Permet rebre informació de la placa *Arduino* al component.
- **GND**: Connectat al GND del *Arduino* (*ground* o terra).
- **VCC**: Connectat a 5 V
- **KEY**: Si és posa a HIGH (3,3 V), abans de connectar el VCC, permet entrar en el mode AT del dispositiu. Aquest mode permet variar característiques i opcions del dispositiu: el mode esclau/màster, el nom que apareix, la contrasenya d'aparellament, etc. [8]

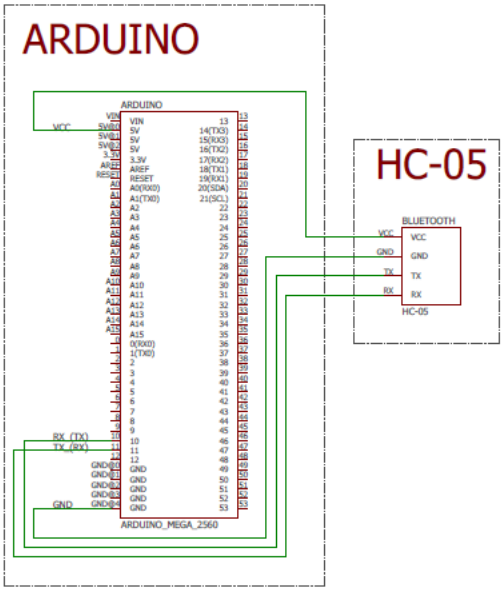

Fig. 6.5. Esquema de connexions entre el mòdul i l'*Arduino*. Font pròpia.

Per connectar aquest component a la placa *Arduino* cal seguir l'esquema presentat a continuació a la Fig. 6.5. Aquest i els següents esquemes s'han realitzat amb el programa *EAGLE* d'*Autodesk* [9].

Un cop fetes les connexions, s'encén el dispositiu i s'enllaça amb el telèfon mòbil. El LED del mòdul ens indica que s'ha emparellat correctament. Amb això fet, ja es té un canal de comunicació establert amb la placa *Arduino* i es pot començar a enviar la informació que convingui.

6.1.1.3. Shield VNH2SP30

Les *shield* són components, sovint amb un format de placa, que es poden connectar a una placa *Arduino* per ampliar les seves capacitats o prestacions. Es fa ús d'aquests components perquè són fàcils de muntar i barats de produir. Cal afegir que moltes *shield* estan dissenyades per ser muntades sobre la mateixa placa *Arduino* però aquestes normalment utilitzen les dimensions de la placa *Arduino* UNO [10].

En el nostre cas es va fer servir una *shield* que ha permès controlar els motors de 12 Volts que incorpora el nostre robot (5.1). Aquests motors no es poden connectar directament a la placa *Arduino* donat l'alt voltatge i amperatge que requereixen. Per solucionar aquest problema es va adquirir aquest component que, connectat a l'*Arduino*, permet regular la intensitat que circula pels motors (la velocitat d'aquests), el sentit (invertint la diferencia de potencial) i tot això sense posar en perill la placa *Arduino*.

A la següent figura (Fig. 6.6) hi ha un mapa amb les connexions necessàries.

Com es veu el disseny i la nomenclatura dels pins es molt semblant al d'una placa Ardunio. Igualment, hi ha pins de la mateixa *shield* que no cal ocupar, doncs són pins lliures. A continuació es detallen algunes de les connexions requerides entre l'*Arduino* i la *Shield.*

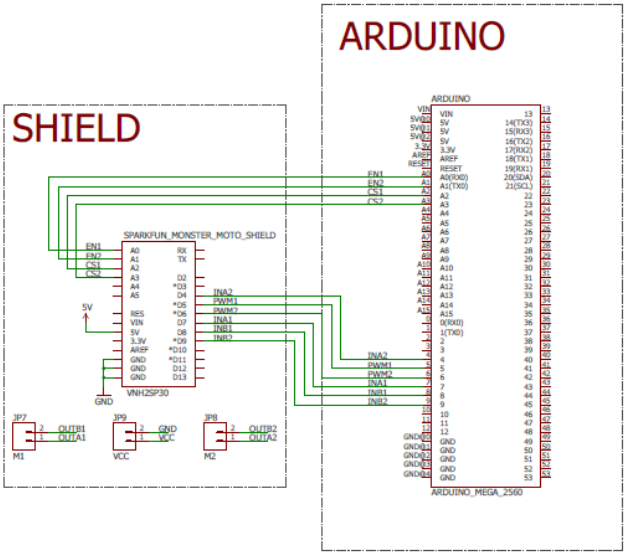

Fig. 6.6. Esquema de connexions entre la *shield* i l'*Arduino*. Font pròpia.

- **VCC**: Connectat a 5 V. Aquesta entrada permet alimentar la *shield* internament, no els motors.
- **GND**: Connectat al GND del *Arduino* (*ground* o terra).
- **IN**: Són un total de quatre pins, dos per cada motor, que controlen l'estat de cada motor a través del voltatge que els hi arriba. Com que hi ha dos pins per cada motor, tenim un total de quatre possibilitats d'estat.
    - Si els dos pin estan amb voltatge *low* tenim curtcircuit a Vcc.
    - Si tenim un pin amb voltatge *high* i l'altre amb *low* o a la inversa, els motors van en sentit de les agulles del rellotge o a l'inrevés. Aquests són els 2 únics estats correctes per obtenir moviment dels motors.
    - Si tenim els dos pins amb voltatge *high* tenim un curtcircuit amb el terra o *ground*.
- **PWM**: Són dos pins, un per cada motor, que reben un voltatge en forma d'impulsos segons la velocitat requerida. La teoria dels pins PWM queda explicada a l'apartat 6.1.1.1.
- **EN:** Són dos pins, un per cada motor, bidireccionals. Això vol dir que tan poder donar informació a la *shield* com enviar-ne a l'*Arduino* (*input* o *output* respectivament). Aquests pins requereixen estar connectat en mode *pullup* per part de l'*Arduino* [11].

Aquest mode permet atribuir, des de l'*Arduino*, un valor per defecte al pin, en aquest cas *high*. Si es detecta un problema, com l'excés de temperatura o un voltatge elevat, la *shield* canvia automàticament el valor a low. Conseqüentment s'aplica una acció protectora o s'atura el funcionament de la *shield*. En la *datasheet* annexada del component podem trobar tota la informació.

- **CS:** Són dos pins, un per cada motor, que donen informació a l'*Arduino* de la corrent que circula pel motor. Aquests pins de sortida són dispensables pel funcionament bàsic del robot i s'ha decidit prescindir de la seva lectura.

Cal tenir en compte que aquest dispositiu fa de pont entre l'*Arduino* i els motors, així que ha de tenir entrades per rebre el voltatge i per subministrar-lo als motors. Concretament trobem:

- **VCC**: Es diferencia de les entrades anteriorment descrites, ja que la connexió no és a través de pins si no de borns on hi arriben cables. No s'ha de confondre amb l'entrada de 5 Volts esmentada en la llista anterior. Aquesta entrada permet alimentar els motors.

- **OUT:** Es diferencia de les entrades anteriorment descrites, ja que la connexió no és a través de pins si no de borns on hi arriben cables. Són les dues sortides que serveixen per alimentar els motors. La corrent que circula per aquestes sortides depèn directament del que la *shield* decideix subministrar i, com ja s'ha esmentat, aquesta decisió sorgeix de les senyals que l'*Arduino* envia a través dels pins anteriorment descrits.

A la Fig. 6.7 podem veure tot el sistema de connexions muntat entre la *shield*, l'*Arduino* i els motors.

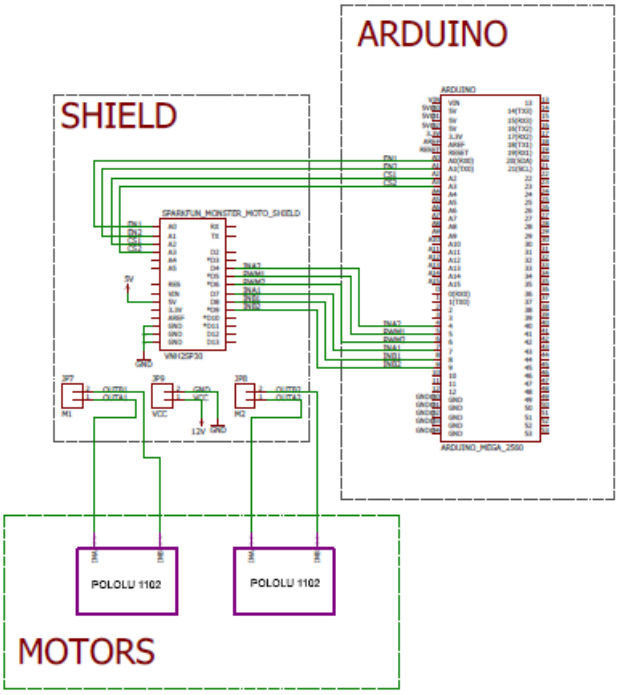

Fig. 6.7. Esquema de connexions entre la *shield,* l'*Arduino* i els motors. Font pròpia.

6.1.1.4. Esquema final de la instal·lació electrònica amb interruptors, cables i altres components de suport.

A la Fig. 6.8 podem veure la visió general del circuit ja muntat en la plataforma principal.

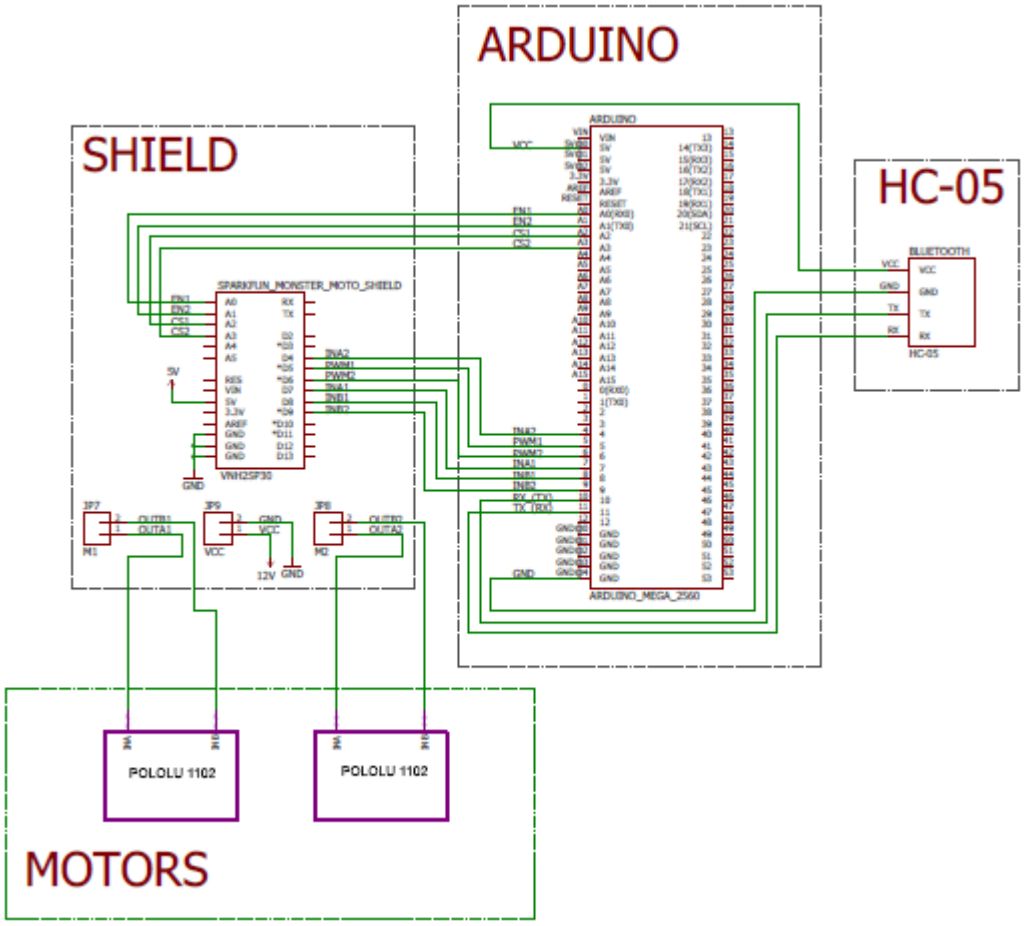

Fig. 6.8. Esquema amb totes les connexions. Font pròpia.

Cal afegir que per unir els components es van utilitzar dos tipus de cables: un de més fi, que és bastant comú en els circuits fets amb *Arduino* i que permet la conducció d'uns pocs ampers i, un altre més gruixut, que assegura el correcte funcionament d'aquells elements que tenen un voltatge i/o amperatge superior; com els motors o les bateries. Com hem vist a la figura anterior totes les connexions entre *Arduino* – *Shield* i *Arduino* – *Bluetooth* són amb els cables més fins, mentre que les connexions entre Bateries – *Arduino*, Bateries – *Shield* i *Shield* – Motors estan fetes amb cables més gruixuts i trenats entre ells per evitar interferències amb la resta de components.

Altres components són:

- Les regletes transparents, que serveixen per empalmar els cables d'alt amperatge.
- Un interruptor, reutilitzat d'impressora, que serveix per encendre o apagar el circuit.
- Dues regletes de dos borns soldades a la *Shield* que serveixen per a poder connectar cables còmodament. Aquestes permeten ajuntar de manera ferma però temporal els cables i resisteixen un amperatge de fins a 16 A.
- Unes forquilles que es van soldar en els borns dels motors per tal d'empalmar-hi els cables corresponents.
- Un connector circular, que permet connectar l'*Arduino* directament als 12 Volts que subministra la bateria.
- Un conjunt de cargols, femelles i separadors hexagonals mascle-femella, disposats de tal manera que permeten mantenir els diferents components electrònics cargolats a la plataforma però elevats uns pocs mil·límetres.

En la Fig. 6.9 es poden veure tots aquests components amb més detall.

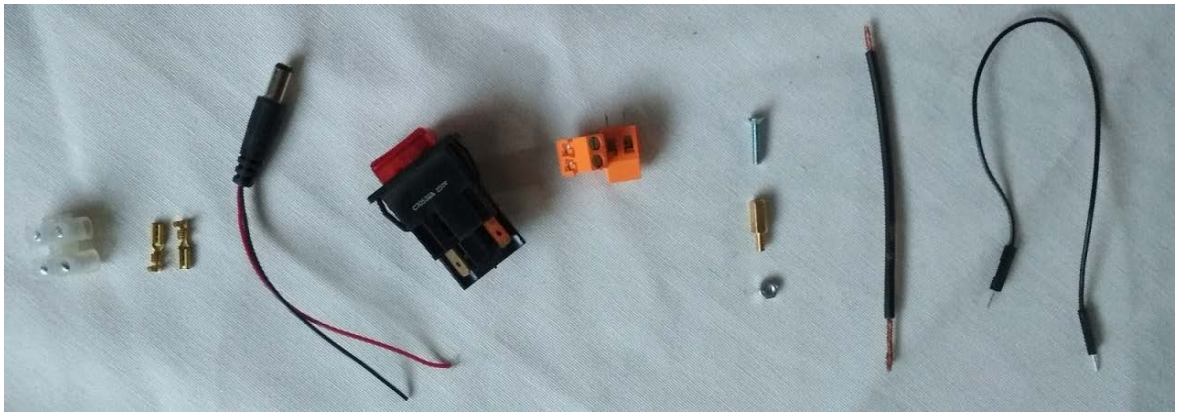

Fig. 6.9. Per ordre: regleta transparent, forquilles, connector circular, interruptor, regletes de dos borns, cargols amb separadors, cable gruixut i prim. Font pròpia.

Es va intentar fer un circuit el més optimitzat possible i compacte per evitar les possibles interferències entre components. A la següent pàgina s'adjunta el disseny del circuit complet, sense incorporar regletes i interruptors, ja que són components opcionals i que no aporten informació essencial al sistema.

6.1.2. Motius Arduino

Actualment, en el mercat existeixen molts tipus de marques i comunitats que contínuament generen noves plaques i components que podrien ser un substitut compatible de la placa *Arduino* utilitzada.

La realitat de l'elecció és ben simple. Es disposava ja d'una placa *Arduino MEGA* que no tenia un ús concret, i ja s'havia treballat i programat anteriorment amb el programari d'aquesta marca. Així mateix, són unes plaques molt assequibles i que donen pocs problemes per treballar-hi a nivell usuari. És cert que hi ha altres propostes en el mercat i que altres dispositius haurien estat més econòmics o òptims per un sistema com el nostre però, amb les primeres aproximacions que es van fer l'inici del projecte, ja es va veure que era viable la programació de tot el codi amb el llenguatge que proposa *Arduino* i es va decidir continuar per aquesta via. En futures millores és podria valorar el fet de fer canvis en aquest aspecte o provar noves tecnologies en altres zones com el cap, si es volgués implementar noves funcionalitats.

6.2. Software

Amb els components electrònics i elèctrics connectats correctament cal descriure quin ha de ser el seu comportament. A la pràctica aquesta *descripció* és fa a través d'un programa que l'*Arduino* executa. Aquest programa o software permet enviar els senyals elèctrics que són necessaris a cada component perquè aquests tinguin un comportament concret. El programa es guarda a la memòria de la nostre placa *Arduino* i el converteix en el cervell de tot el sistema.

6.2.1. Arduino – IDE

Com s'ha introduït en l'apartat 6.1.1.1, la companyia que fabrica les plaques *Arduino* també disposa d'un entorn de programació gratuït que serveix per escriure, depurar, compilar i pujar un codi a la placa *Arduino* a través d'un cable USB o connexió indirecta amb un ordinador (*WIFI*, *Bluetooth*, etc.). Tal i com apareix en el glossari, un entorn que agrupa aquestes funcions s'anomena *IDE* o entorn integrat de desenvolupament.

Actualment, *Arduino* disposa d'una versió en línia i una versió instal·lable d'aquest entorn. En aquest cas es va fer servir la versió 1.6.9 instal·lada sobre Windows 10 de 64 bits. Com també s'ha comentat anteriorment, el llenguatge de programació que entén l'*Arduino* està basat en C i inclou funcionalitat de C++. Per facilitar la lectura del codi font, l'entorn inclou un conjunt de colors que agrupen funcionalitats de la mateixa categoria. Per exemple, tots els tipus de variables surten en color blau, les funcions que reconeix o que formen part d'alguna

llibreria en taronja i els condicionals, bucles i directives en verd. Una directiva és un tros de codi que no s'executa si no que dóna informació al compilador per saber com actuar [12].

A la Fig. 6.10 en veiem un exemple.

```
#include <Mouse.h>

#define Click 0

int state = 0;

void setup() {}

delay(670);
```

Fig. 6.10. Mostra dels diferents colors que estableix el mateix entorn segons el seu criteri. Font pròpia.

En els annexos d'aquesta memòria s'hi pot trobar el codi font utilitzat per manipular el robot des de l'aplicació mòbil a través del mòdul *Bluetooth*. Aquest codi està explicat i, a més a més, a continuació es repassaran totes aquelles funcionalitats que té incorporades. Tanmateix, aquesta memòria no pretén ser una guia de com aprendre a programar amb *Arduino* i s'ha considerat que explicar detalladament funcions que no s'han utilitzat o estructures que qualsevol llenguatge de programació incorpora, podia fer desviar el treball de l'objectiu de mostrar i explicar el nostre programa. Revisant la documentació que ofereix gratuïtament i en línia el lloc web d'*Arduino*, s'ha vist que hi ha un racó exclusiu que documenta les referencies del llenguatge. És una bona eina per anar ampliant coneixements i resoldre els dubtes que puguin sorgir sobre el llenguatge [13].

El que si que cal tenir en compte és que, per comentar sobre el codi i que el compilador no ho entengui com a codi a depurar i compilar, hi ha 2 mètodes. Un és afegir al principi de frase dos barres diagonals // i escriure la frase corresponent a continuació. L'altre serveix per comentar un bloc i un exemple seria així: */* Text comentat en bloc */.* En les figures que mostren el codi es veurà més clarament l'ús i els beneficis de cadascuna.

6.2.2. Funcionalitats a les que respon

Abans d'entrar en detall amb tot el codi font, es vol explicar a grans trets el funcionament del programa quan s'executa a l'*Arduino*.

En el moment que l'*Arduino* s'encén el programa s'inicia i el primer que fa és establir connexió amb el mòdul *Bluetooth* i començar a rebre i enviar informació. El mòdul *Bluetooth* està configurat per esperar una connexió o aparellar-se amb un nou dispositiu. Quan l'aparellament és correcte, l'*Arduino* envia un missatge al emissor, avisant que tot ha estat un èxit. Si el mòdul *Bluetooth* rep alguna informació, l'envia a l'*Arduino* i aquest la processa. Aquesta informació només poden ser caràcters alfanumèrics que poden indicar la direcció, si són lletres, o la velocitat, si són números. Un cop hi ha nova informació, l'*Arduino* sap quin moviment s'espera i actua enviant als pins digitals o analògics corresponents la informació que sigui necessària. Aquests pins comuniquen amb la *shield* que actua en consonància i dona un cert amperatge als motors.

Com s'ha comentat anteriorment, *Arduino* disposa d'una documentació extensa sobre el seu llenguatge. Aquesta està classificada segons si tracta sobre variables, funcions o estructures. Per l'explicació del codi font seguirem el mateix esquema però diferenciant les funcionalitats que porta definides el llenguatge i les creades per nosaltres.

6.2.2.1. Estructures

Hi ha dues estructures bàsiques i necessàries que automàticament afegeix l'entorn en iniciar un nou projecte. Aquestes són *setup()* i *loop().*

L'estructura *setup()* es la primera funció que s'inicia en encendre l'*Arduino* o fer un *reset*. Aquesta inicialitza les variables, els estats dels pins i comença a utilitzar les llibreries definides.

En el nostre programa, aquesta estructura inicia la comunicació amb l'*Arduino* i amb el *Bluetooth*. Conté dos bucles fets amb un *for,* que a través de la funció pinMode() estableix el mode d'un pin concret. Els pins poden estar en mode INPUT (entrada), OUTPUT (sortida) i INPUT_PULLUP. També posa a voltatge baix (LOW) els pins d'estat A i B de la *shield* dels dos motors. A més a més, actualitza els valors dels registres TCCR3B i TCCR4B. Això va ser una línia de codi trobada juntament amb el tutor del treball per augmentar la freqüència del PWM dels pins digitals 5 i 6. Sense això el PWM no donava els polsos correctament i el motor tenia un mal funcionament [14].

Finalment, envia un missatge al mòdul *Bluetooth* avisant que la connexió està establerta. Podem veure les línies de codi corresponents a la Fig. 6.11.

```
void setup() // Estructura predefinida
{
  Serial.begin(38400); //Inicialització del port de comunicació Arduino
  BTSerial.begin(38400); //Inicialització del port de comunicació Bluetooth

  /* Aquesta acció es repeteix fins que la variable i que inicialment té un valor de 0
   passi a tenir un valor superior a 2 incrementant-la a cada cicle 1 unitat*/
  for (int i=0; i<2; i++)
  {
    //Estableix el mode del pin com a entrada o sortida (INPUT o OUTPUT)
    pinMode(inApin[i], OUTPUT); //En aquest cas el mode és OUTPUT
    pinMode(inBpin[i], OUTPUT);
    pinMode(pwmpin[i], OUTPUT);
    pinMode(enpin[i],  INPUT_PULLUP); //En aquest cas el mode és INPUT_PULLUP
    pinMode(2,  INPUT_PULLUP);
  }
  for (int i=0; i<2; i++)
  {
    digitalWrite(inApin[i], LOW); // Escriu un valor digital en un pin
    digitalWrite(inBpin[i], LOW);
  }
  TCCR3B = TCCR3B & B11111000 | B00000010; // Actualitza el valor del registre TCCR3B
  TCCR4B = TCCR4B & B11111000 | B00000010;

  //pinMode(12, OUTPUT);
  SendBluetooth("Comunicació establerta"); // Crida a la funció SendBluetooth()
}
```

Fig. 6.11. Estructura *setup()* del codi font. Font pròpia.

L'estructura *loop()* és l'estructura que es crida quan *setup()* ha acabat. És el bucle principal del programa i dintre seu succeeixen totes les crides a funcions i decisions concretes. L'*Arduino* necessita, recursivament, executar aquest bucle per funcionar.

En el programa creat, és el bucle que revisa si hi ha nova informació rebuda, cridant a la funció *check()*. Posteriorment, entra en un bucle fet amb la paraula reservada *while,* per cada caràcter que pot rebre i segons quin sigui envia a la funció *setSpeeds()* les velocitats corresponents de cada motor. Aquest bucle no s'acaba mai. Veiem una imatge d'aquesta estructura en la Fig. 6.12.

```
void loop() // Estructura predefinida
{
  det = check(); // Actualitza la variable DET amb el valor de retorn de la funció check()

  while (det == 'F')   // F, moviment endavant
  {setSpeeds(vel,vel);det = check();} // Crida la funció setSpeeds() amb la velocitat que marca VEL
                                      // Actualitza la variable DET amb el valor de retorn de la funció check()
  while (det == 'B')   // B, moviment endarrere
  {setSpeeds(-vel,-vel);det = check();} // Crida la funció setSpeeds() amb la velocitat negativa que marca VEL

  while (det == 'L')   // L, moviment de les rodes a l'esquerre
  {setSpeeds(vel,-vel);det = check();}

  while (det == 'R')   // R, moviment de les rodes a la dreta
  {setSpeeds(-vel,vel);det = check();}

  while (det == 'I')   // I, anar cap a la dreta endavant
  {setSpeeds(float(vel/2.0),vel);det = check();}

  while (det == 'J')   // J, anar cap a la dreta endarrere
  {setSpeeds(float(-vel/2.0),-vel);det = check();}

  while (det == 'G')   // G, anar cap a l'esquerre endavant
  {setSpeeds(vel,float(vel/2.0));det = check();}

  while (det == 'H')   // H, anar cap a l'esquerre endarrere
  {setSpeeds(-vel,float(-vel/2.0));det = check();}

  while (det == 'S')   // S, parar
  {setSpeeds(0,0);det = check();} // Crida la funció setSpeeds() amb la velocitat a 0
}
```

Fig. 6.12. Estructura *loop()* del codi font. Font pròpia.

Com passa en molts llenguatges, *Arduino* permet importar i exportar llibreries pròpies o externes. Per poder incloure aquestes llibreries en el projecte es va utilitzar la directiva *#include <XXX.h>,* on XXX és el nom de la llibreria escrita en C. Hi ha una altra directiva que va ser utilitzada en el codi font anomenada *#define,* que permet assignar un nom a un valor constant, fent-lo més intel·ligible per una persona que el llegeix.

En el codi es va importar la llibreria SoftwareSerial, que és pròpia d'*Arduino* i permet fer servir les funcions que defineixen la comunicació externa. També es van definir 4 constants: *BrakeVCC, CW, CCW i BrakeGND.* Aquestes constants fan referencia a un valor i en realitat expressen les 4 possibilitats que té la *shield*: curtcircuit a VCC, gir en sentit horari, gir en sentit antihorari i curtcircuit a *ground*. A més a més, hi ha una línia comentada que defineix *CS_THRESHOLD* però, com ja s'ha explicat en l'apartat 6.1.1.3, el pin de control és totalment innecessari i en aquest cas, aquesta constant establia un llindar màxim pel pin *CS*. Tot això és pot veure en la Fig. 6.13.

```
/* Aquesta llibreria és indispensable per utilitzar les funcions Serial
 de l'Arduino. Aquestes són les funcions de comunicació.*/
#include <SoftwareSerial.h> // Importa la llibreria SoftwareSerial

#define BRAKEVCC 0 // Defineix la variable BRAKEVCC amb el valor 0
#define CW   1 // Defineix la variable CW amb el valor 1
#define CCW  2 // Defineix la variable CCW amb el valor 2
#define BRAKEGND 3 // Defineix la variable BRAKEGND amb el valor 3

//#define CS_THRESHOLD 100
```

Fig. 6.13. Directives en el codi font. Font pròpia.

Es contempla dins d'estructures qualsevol operació aritmètica, operació de comparació, operació booleana o altres operacions. Una operació booleana és aquella que té un resultat que pot prendre exclusivament el valor de *false* o *true*, també escrit com 0 o 1 [15].

A més a més, hi ha estructures de control com són les estructures condicionals *if...else* o els bucles fets amb *for* o *while* que també estan en aquesta categoria. Tot i que aquestes estructures van ser utilitzades en el codi, qualsevol persona que programi a un nivell bàsic les haurà utilitzat. En cas de no tenir cap noció de programació pot ser d'ajut consultar el lloc web que hi ha en el peu de l'anterior pàgina. Quan ens trobem davant d'aquestes estructures dins de les funcions pròpies s'explicarà el funcionament del conjunt però detallant totes les casuístiques que hi poden haver i, en certa manera, donant a conèixer la utilitat d'aquestes.

6.2.2.2. Variables

Les variables són una part important del programa. Contenen dades que poden o no ser modificades pel propi *Arduino* i que interpretades pel programa permeten actuar de certa manera. Per exemple, podem establir una variable que prengui dos valors (0 i 1) i que indiqui l'estat d'un component (apagat i obert). Depenent del valor de la variable l'*Arduino* actuarà d'una manera o d'una altra i la variable s'haurà d'actualitzar si l'estat del component canvia.

Per parlar de variables primer parlarem dels tipus que aquestes poden tenir. En general hi ha una llarga llista de tipus però especificarem els més comuns:

- **Byte**: Ocupa 8 bits i conté un numero del 0 al 255
- **Boolean** : Ocupa un byte de memòria i pot contenir un de dos valors – true o false.
- **Char**: Ocupa un byte de memòria i pot contenir un únic caràcter. S'escriu amb

cometes simples.

- **Int**: Permet guardar números no decimals amb signe. Depèn del processador pot guardar un numero més o menys gran, així doncs pot ocupar de 2 a 4 bytes.
- **Float**: Permet guardar números decimals amb signe. Ocupa 4 bytes de memòria i té una precisió de 6 o 7 decimals.
- **String**: Ocupa els bytes necessaris per contenir la paraula. S'escriu amb cometes dobles i sempre es defineix com un *array* de caràcters (*char*).
- **Array**: És una col·lecció de variables accessibles a través del seu índex o posició. S'escriuen entre claus { . . . }.

A més a més, *Arduino* té un conjunt de variables constants predefinides i que juntament amb les funcions que l'entorn té incorporades, entre d'altres coses, faciliten la comunicació amb els pins. L'exemple més fàcil seria el de les variables *true* o *false* que corresponen als valors 1 o 0 respectivament, però de forma escrita. Un altre exemple apareix en la Fig. 6.11 del punt d'estructures on apareixen les variables constants INPUT, OUTPUT i INPUT_PULLUP. Si cal més informació hi ha tota una documentació sobre variables constants [11].

En el nostre programa, després de l'ús de les directives anteriorment descrites però abans de l'estructura *setup(),* hi ha la definició de totes les variables globals que el codi farà servir i, si s'escau, el seu valor inicial. Una variable global és una variable que conserva el seu valor i pot ser actualitzada per qualsevol funció que la precedeixi. En la Fig. 6.14 veiem aquest fragment del codi comentat. La definició del pin *CS* de la *shield* està desactivada, ja que no s'utilitza; això se sap perquè aquesta està comentada íntegrament. També cal esmentar que la inicialització de *BTSerial* és fa a través de la llibreria *SoftwareSerial,* que la dota d'un tipus concret amb unes característiques i funciones heretades descrites per aquesta llibreria.

```
SoftwareSerial BTSerial(10, 11); // Defineix els pins de comunicació TX i RX del mòdul Bluetooth
int inApin[2] = {7, 4};  // INA: Pins d'entrada per girar en el sentit horari [DIGITAL]
int inBpin[2] = {8, 9}; // INB: Pins d'entrada per girar en el sentit antihorari  [DIGITAL]
int pwmpin[2] = {5, 6}; // PWM Pins entrada [DIGITAL]
int enpin[2] = {A0, A1}; // EN: Pins d'estat dels interruptors de sortida (enable) [ANALOG]

//int cspin[2] = {A2, A3}; // CS: Current sense analog input [ANALOG]

char det = 'S'; // Inicialitza la variable global det com un char amb el valor 'S'
int vel = 400; // Inicialitza la variable global vel com un int amb el valor 400
char dataIn = 'S'; // Inicialitza la variable global dataIn com un char amb el valor 'S'
char determinant; // Inicialitza la variable global determinant com un char i sense valor
int SecureStop = 0; // Inicialitza la variable global SecureStop com un int amb el valor 0
```

Fig. 6.14. Variables globals definides en el codi font. Font pròpia.

La inicialització dels pins està definida com una *array* de *int,* és a dir una col·lecció de valors enters (exemple *inAin[2] = {7, 4}*). Si escrivíssim aquesta inicialització en un llenguatge intel·ligible diria: la variable *inAin* és una llista amb 2 espais que pot contenir valors enters. La trampa és que en aquest cas estem definint pins i, com bé sabem, *Arduino* té pins analògics i digitals i alguns tenen numeracions superposades. És per això que en la definició de la variable *enpin* s'utilitza el caràcter A abans del número enter; serveix per fer saber al *Arduino* que estem parlant d'un pin analògic. Informació que posteriorment farà servir la funció *pinMode()*.

6.2.2.3. Funcions

Hi ha dos tipus de funcions a explicar: les que integra *Arduino* i les que s'han generat de zero. Primerament parlarem de les funcions que integra el mateix entorn.

- *pinMode(pin, mode)*: Configura un pin específic per comportar-se com una entrada o una sortida. Els modes vàlids són les variables constants *INPUT, OUTPUT, INPUT_PULLUP*

- *digitalWrite(pin, valor)*: Escriu un valor alt o baix de voltatge a un pin digital específic. Aquest valor es representa amb les variables constants *HIGH* i *LOW.*

- *analogWrite(pin, valor)*: Escriu un valor analògic de voltatge a un pin analògic específic. Aquest valor ha d'estar entre 0 i 255.

- *Serial.begin(velocitat):* Inicia el port sèrie per la transmissió de dades. Un port sèrie és un tipus de maquinari de comunicació sèrie que permet la transmissió o la recepció de bits un darrere de l'altre [16]. La velocitat es mesura en baud per segon.

- *Serial.available()*: Obté el número de bytes disponibles per llegir en el port sèrie.

- *Serial.read()*: Llegeix les dades que entrants del port sèrie.

- *Serial.write(msg)*: Envia un conjunt de dades (caràcters) a través del port sèrie. La variable msg conté aquestes dades.

- *map(valor, inicialMínim, inicialMàxim, finalMínim, finalMàxim)*: Redimensiona un valor d'una escala a una altra. L'escala inicial té un inicialMínim i un inicialMàxim, la nova escala té un valor finalMínim i un valor finalMàxim.

Totes aquestes funcions es troben sovint dins d'altres estructures més grans. Al final, les funcions pròpies permeten fer un conjunt d'operacions i cridar a aquestes funcions predefinides que poden influir sobre els pins de l'*Arduino*. En el codi font de l'annex hi ha totes les funciones pròpies comentades àmpliament. Totes elles disposen de la següent informació: nom de la funció, variables requerides, retorn de la funció i descripció d'aquesta. Comentar-les una per una fora del seu context no té molt sentit. És per això que només s'adjunta la Fig. 6.15, on es pot veure un exemple de funció totalment comentada.

```
/*
 * Nom: motorGo()
 * Variables requerides: motor (byte), direct (byte) i pwm (byte).
 * Retorn de la funció: Cap
 * Descripció: Donat l'identificador d'un motor (motor), el sentit de gir (direct) i
 *             la velocitat en format byte (pwm). Permet establir aquestes característiques
 *             al motor concret escrivint aquesta informació en els pins.
 */
void motorGo(uint8_t motor, uint8_t direct, uint8_t pwm)
{
  if (motor <= 1) // Si el valor de MOTOR és inferior o igual a 1, valida el condicional
  {
    if (direct <=4) // Si el valor de DIRECT és inferior o igual a 1, valida el condicional
    {
      if (direct ==1) // Si el valor de DIRECT és igual a 1, valida el condicional
      {
        digitalWrite(inApin[motor], HIGH);
        digitalWrite(inBpin[motor], LOW);

      }
      else if(direct ==2) // Si el valor de DIRECT és igual a 2, valida el condicional
      {
        digitalWrite(inApin[motor], LOW);
        digitalWrite(inBpin[motor], HIGH);
      }
      else // En cas que DIRECT sigui diferent a 1 o 2 (cas impossible amb un bon funcionament).
      {
        digitalWrite(inApin[motor], LOW);
        digitalWrite(inBpin[motor], LOW);
      }
      analogWrite(pwmpin[motor], pwm);
    }
  }
}
```

Fig. 6.15. Extracte codi font amb el comentari de la funció *motorGo()*. Font pròpia.

6.2.3. Actualització versions

Actualment, per anar pujant les noves versions la placa *Arduino* és connecta a través d'un cable USB tipus A/B a l'*Arduino* a un ordinador i executant l'aplicació que ofereix *Arduino*, es verifica, compila i puja el codi font. Un cop pujat a l'*Arduino* no hi ha manera d'extreure'l, és a dir, no podem analitzar quin codi font té la nostra placa, només podem re-escriure-la amb un nou codi. Tot i que s'ha llegit documentació per fer aquest procés sense cables físics, encara no s'ha implementat aquesta funcionalitat i sempre es requereix d'accés a la placa i d'un ordinador físic per actualitzar o corregir el codi incorporat.

6.3. Aplicació mòbil

La necessitat de controlar el robot a distància va ser un dels pilars del projecte. Hi havia múltiples opcions per solucionar aquest problema; des de comprar un *joystick* o palanca de control sense fils que permetés la comunicació amb l'*Arduino*, fins a fer ús d'un ordinador per comunicar-se. La comunicació seria a través d'algun canal com el *Bluetooth*, la WIFI o els infrarojos. Tanmateix, es va decidir utilitzar un dispositiu que, actualment, tothom porta sempre a sobre: el mòbil. Ara per ara, la majoria de models tenen implementades si no totes, la majoria d'aquestes tecnologies de comunicació. Aquesta opció té un cost menor i permet que qualsevol mòbil esdevingui un comandament.

Si es parla d'aplicacions mòbils primer cal esmentar els dos sistemes operatius més utilitzats avui dia pels telèfons intel·ligents: Android, que pertany a Google, i iOS, que pertany a Apple. Un sistema operatiu és un conjunt de programes que controlen el funcionament d'un ordinador [17].

El procés per crear aplicacions per cadascun d'ells és molt diferent; començant pel llenguatge de programació en el que es basen i passant pels tràmits i taxes que cal pagar si es volen acabar publicant. En aquest aspecte Apple és molt més corporatiu i tancat. Per exemple, la botiga d'aplicacions d'iOS és molt més restrictiva ja que les llicencies dels desenvolupadors són de pagament i només aquelles aplicacions que compleixen uns requisits imposats per la companyia són acceptades. Les eines que proporciona Apple per desenvolupar aplicacions natives solen funcionar exclusivament en sistemes operatius iOS d'ordinador. Pel contrari, Android dóna molta més llibertat als desenvolupadors, posant a l'abast dels usuaris una botiga plena d'aplicacions i la possibilitat de crear aplicacions a través de software lliure.

Abans de continuar, es definiran alguns conceptes que cal tenir clars. Primerament, una aplicació nativa és aquella aplicació que està programada per funcionar en un sistema

operatiu específic. Actualment, és comú que les aplicacions no siguin natives per poder-les reutilitzar en altres sistemes operatius com iOS, WindowsPhone, etc. D'aquesta manera s'estalvia crear repetidament una mateixa aplicació en diferents llenguatges de programació; el desavantatge és que aquesta no estarà optimitzada pel dispositiu que utilitzarem.

En el moment que es va plantejar la creació d'una aplicació mòbil, el coneixement i experiència en aquest camp era nul. És per això que, en un inici, es va valorar fer servir una aplicació ja creada de la botiga de Google. Aquesta en concret ja es proposava en la guia [1] que anteriorment s'ha citat. Més endavant, si es veia la possibilitat i facilitat per crear una aplicació pròpia, es tiraria endavant.

6.3.1. Aplicació inicial

L'aplicació inicial que es va fer servir era una aplicació gratuïta de la botiga *Google Play* anomenada "*Arduino Bluetooth RC Car*". Aquesta, en obrir-se, encén el *Bluetooth* i et mostra una pantalla amb fletxes que es poden polsar. A la Fig. 6.16 es mostra visualment el procés.

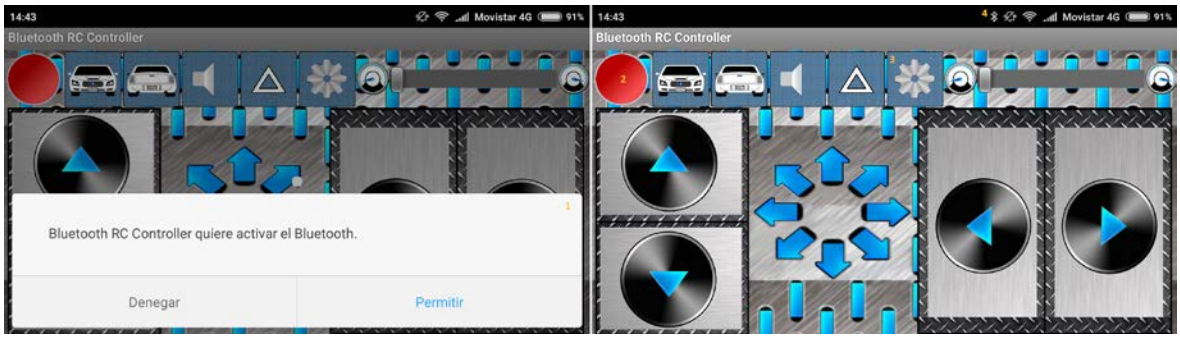

Fig. 6.16. Captures de pantalla de l'aplicació inicial: 1. Activació del *Bluetooth*, 2. Estat de la connexió, 3. Icona d'opcions i 4. Icona *Bluetooth* habilitat en el mòbil. Font pròpia.

Des d'aquí, per connectar-se amb el robot, és necessari anar a opcions (icona de l'engranatge) i fer clic a *Connect to Car.* Si ja es tenia el dispositiu *Bluetooth* del mòbil enllaçat amb el del robot apareix el seu nom; BB8 en aquest cas. Tot això es veu a la Fig.

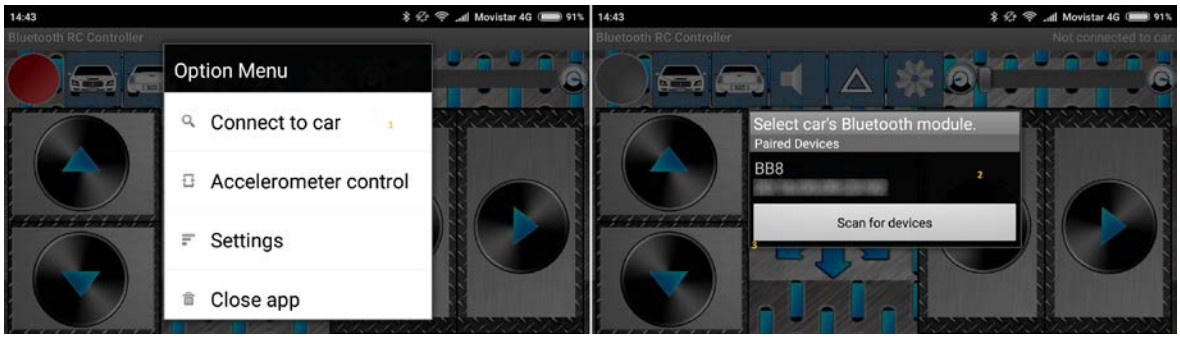

Fig. 6.17. Captures de pantalla de l'aplicació inicial utilitzada: 1. Menú d'opcions, 2. Selecció dels dispositius enllaçats i 3. Nova cerca de dispositius *Bluetooth*. Font pròpia.

6.17.

Com veiem a la Fig. 6.18, un cop connectat, la icona que mostra l'estat de la connexió es torna verda i apareix un missatge de connexió satisfactòria. Ara ja, si es clica sobre les fletxes o el selector de velocitat, s'envia la informació a l'*Arduino*.

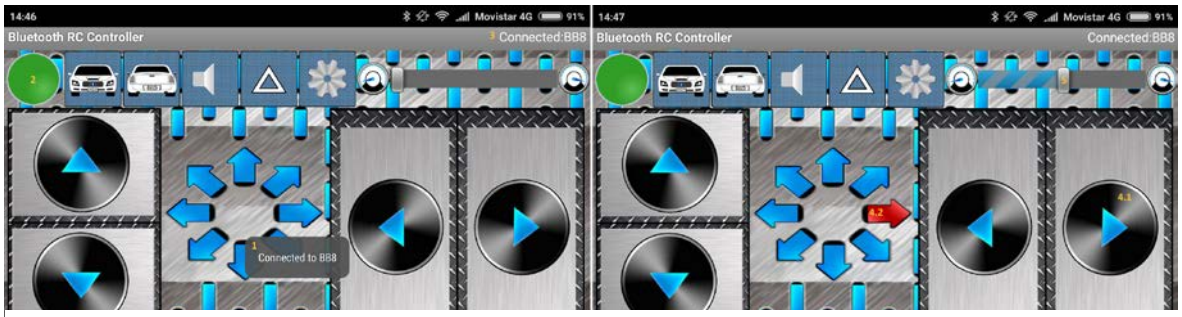

Fig. 6.18. Captures de pantalla de l'aplicació inicial utilitzada: 1. Missatge confirmació connexió, 2. Estat de la connexió, 3. Nom del dispositiu al que s'ha connectat, 4.1. Polsador fletxa direcció dreta, 4.2. Indicador direcció polsada i 5. Selector de velocitat. Font pròpia.

Les fletxes es poden combinar per anar, per exemple, endavant a la dreta. Aquesta aplicació també disposa d'uns botons extres per obrir les llums del davant, les de darrere, tocar el clàxon, etc. No és d'estranyar, ja que és una aplicació per controlar un cotxe.

Tenint aquesta aplicació funcionant, es va analitzar quins missatges enviava al receptor; l'*Arduino* en aquest cas. En fer-ho, es van poder programar els condicionals i anar creant el codi presentat en l'apartat 6.2. Per exemple, es va observar que en fer clic a la fletxa que indicava cap amunt, l'*Arduino* rebia una 'F' (de *forward* en anglès). Es van anar analitzant així totes les possibilitats i, paral·lelament, es va anar fent el codi.

Un cop es va obtenir una versió del codi d'*Arduino* que responia correctament als estímuls de l'aplicació, es va valorar crear-ne una pròpia. Aquesta podia tenir un disseny més específic pel projecte, permetia ampliar les funcionalitats si era necessari i es podia programar de tal manera que envies la mateixa informació en clicar els components, per evitar canviar també el codi *Arduino* i poder tenir sempre, per si de cas, l'aplicació inicial operativa.

6.3.2. Anàlisi del software per crear-la

El principal problema de generar una aplicació pròpia era la desconeixença del llenguatge de programació que utilitza *Android*. Aquest és Java, un llenguatge orientat a objectes. Aprendre a programar en Java és una tasca complicada per un període de temps tan curt com el d'aquest treball. Així doncs, feia falta trobar la manera de generar una aplicació utilitzant els coneixements de programació que es tenien.

Actualment, en el mercat hi ha diversos softwares que et fan la feina més fàcil. Simplement programes l'aplicació en un llenguatge concret que coneguis i el programa s'encarrega de compilar l'aplicació per poder-la instal·lar i executar en el mòbil. Òbviament, aquestes aplicacions no són natives i no estan optimitzades, però funcionen igual; pel que es volia no es notava gairebé la diferencia i es podia fer una aplicació molt més còmodament.

Es va dur a terme un petit anàlisi per veure les opcions i se'n van escollir dues: *Kivy* i *Basic for Andorid*. *Kivy* [18] és una llibreria per *Python* [19] que permet desenvolupat aplicacions d'una manera rapida. *Python* és un llenguatge de programació d'alt nivell que s'ensenya durant el grau en tecnologies industrials i amb el qual ja es tenia certa experiència.

Basic for Android o B4A és un software que et permet programar, amb Visual Basic, aplicacions destinades a fer-se servir en un dispositiu Android. Això era una gran sort ja que el mateix llenguatge Visual Basic es va generar per facilitar la programació en ella mateixa i poder simplificar-la. És un llenguatge molt fàcil d'aprendre i, en conseqüència, molt estès.

Es van fer diverses proves i, finalment, es va elegir treballar amb B4A, donada la seva estabilitat i la possibilitat que oferia de crear l'aplicació per iOS en un futur. La configuració de *Kivy* era molt feixuga i sovint fallava per errors que, sense un profund coneixement, no es podien resoldre. Contràriament amb B4A tot eren facilitats: una interfície gràfica per dissenyar l'aspecte de l'aplicació, un fòrum d'ajuda, llibreries externes per afegir components, etc.

6.3.3. B4A

Com ja s'ha comentat, *Basic for Android* (B4A) és un programa que s'ha d'instal·lar a l'ordinador. A la seva web disposa d'una versió de prova (*trial),* que permet utilitzar el *sotfware* per 30 dies i amb limitacions de mida del projecte; i una versió de pagament (*full*), que inclou una llicència perpètua i altres característiques que depenen del tipus de llicència escollida [20].

Inicialment no es tenia cap coneixement de B4A i es va anar aprenent a través de guies i exemples externs. Això va fer que no es pogués entrar en tanta profunditat en totes les funcionalitats i es tingués un coneixement menys ampli i global. Aquest anàlisi no s'estructurarà igual que el del codi *Arduino*: es començarà comentant les funcions i regions que, per defecte, defineix el programari en generar un nou projecte; indispensables per un correcte funcionament de l'aplicació.

Quan es comença de zero una nova aplicació, B4A crea un nou projecte amb una activitat *Main* a dins. El projecte es defineix com el conjunt d'activitats, serveis i mòduls estàtics que formen l'aplicació.

Una activitat és un mòdul que representa una única pantalla amb una interfície d'usuari; permet definir les variables que s'utilitzaran, carregar la interfície que es visualitzarà i definir els processos que permetran generar les diverses funcionalitats [21]. Realment, el concepte d'activitat ve heretat de la programació d'aplicacions amb Java. Java és un llenguatge de programació orientat a objectes, que es va dissenyar per escriure aplicacions un cop i executar-les en qualsevol dispositiu. És el llenguatge de programació utilitzat, en general, per desenvolupar aplicacions mòbil d'Android.

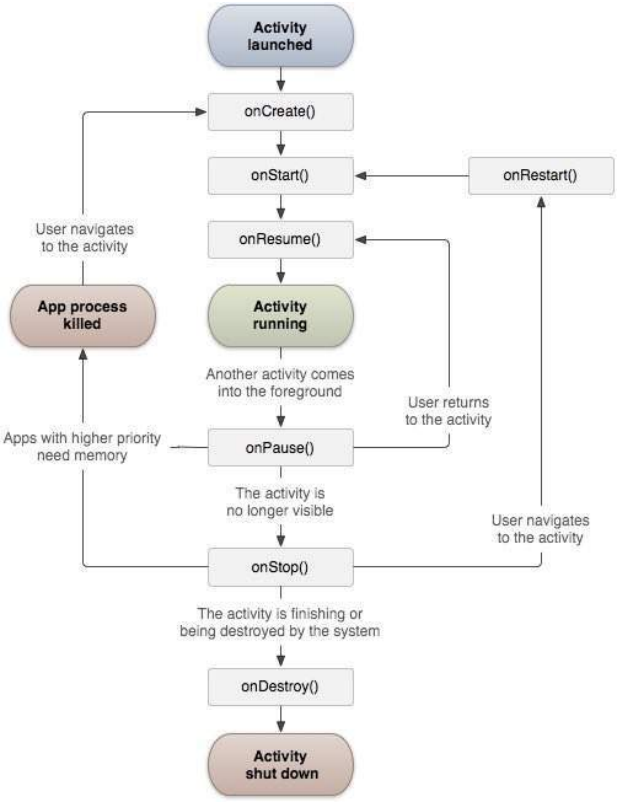

Fig. 6.19. Diagrama de flux d'una activitat amb les funcions executades segons el context. Font: **tutorials**point

Un únic projecte pot contenir més d'una activitat si es volen definir pantalles separades dins d'una aplicació. Cadascuna d'aquestes és independent però, inicialment, l'aplicació sempre cercarà l'activitat que s'anomeni *Main*. Qualsevol activitat ha de contenir, obligatòriament, un conjunt de funcions per estar ben definida i executar-se correctament. Aquestes són: *Activity_Create()*, *Activity_Resume()* i *Activity_Pause().*

En l'esquema de la Fig. 6.19 es presenta un diagrama de flux amb el motiu pel qual es criden cadascuna d'aquestes funcions [22].

Un servei és un mòdul que defineix un conjunt de tasques que s'executen, tot i que l'activitat principal no estigui visible (execució en *background*). Normalment utilitza la barra de notificacions del mòbil per interaccionar amb l'usuari, sent aquest l'únic element visible que genera. En aquest projecte no es va fer servir cap servei i, en conseqüència, no es va aprofundir en la seva utilitat [23].

Un mòdul estàtic de codi (*Static Code Modules*) és un component dispensable en un projecte, que permet definir un conjunt de funcions que poden ser utilitzades per diverses activitats. Aquest es pot reutilitzar en altres projectes. En aquesta aplicació es va utilitzar un mòdul extern anomenat *VSeekbar* creat per un usuari del fòrum de B4A i que serveix per generar barres de lliscants verticals.

B4A també permet generar arxius .bal (*Basic Android Layout*), que contenen el disseny de la interfície gràfica. La interfície gràfica és el conjunt d'element que es podran visualitzar i amb els que l'usuari interaccionarà.

Un cop definits aquests conceptes, es presenta la Fig. 6.20, que mostra l'aspecte de l'activitat *Main* en crear un nou projecte. A continuació, s'aniran descrivint les diferents àrees que hi apareixen.

```
#Region  Project Attributes
    #ApplicationLabel: B4A Example
    #VersionCode: 1
    #VersionName:
    'SupportedOrientations possible values: unspecified, landscape or portrait.
    #SupportedOrientations: unspecified
    #CanInstallToExternalStorage: False
#End Region

#Region  Activity Attributes
    #FullScreen: False
    #IncludeTitle: True
#End Region

Sub Process_Globals
    'These global variables will be declared once when the application starts.
    'These variables can be accessed from all modules.

End Sub

Sub Globals
    'These global variables will be redeclared each time the activity is created.
    'These variables can only be accessed from this module.

End Sub

Sub Activity_Create(FirstTime As Boolean)
    'Do not forget to load the layout file created with the visual designer. For example:
    'Activity.LoadLayout("Layout1")

End Sub

Sub Activity_Resume

End Sub

Sub Activity_Pause (UserClosed As Boolean)

End Sub
```

Fig. 6.20. Aspecte del codi font de l'activitat *Main* d'un nou projecte. Font pròpia.

Una regió és una directiva (*#Region*) que serveix per especificar un bloc de codi que es pot mostrar o ocultar en conjunt [24]. Inicialment, B4A presenta dos tipus de regions: la primera s'anomena *Project Attributes* i permet definir, a l'interior de la regió, el conjunt d'atributs comuns en tot el projecte; la segona s'anomena *Activity Attributes* i permet definir els atributs específics d'aquella activitat. A l'apartat de l'annex que adjunta el codi de B4A comentat, es pot veure quins atributs es van definir per aquesta aplicació.

La comanda *Sub* permet declarar el nom, els paràmetres i el codi d'un procediment que s'executa dins de l'activitat (*sub procedure*) [24]. Els procediments que es generen per defecte són: *Process_Globals()*, *Globals()*, *Activity_Create()*, *Activity_Resume()* i *Activity_Pause().*

El procediment *Process_Globals()* permet definir un conjunt de variables accessibles des de qualsevol mòdul (activitat, servei o mòdul estàtic). Aquestes seran declarades un únic cop, només quan s'iniciï l'aplicació.

El procediment *Globals()* permet definir un conjunt de variables accessibles únicament des de la mateixa activitat. Aquestes es tornaran a declarar cada cop que l'activitat es creï.

Activity_Create() es crida quan l'activitat es crea. Requereix una variable de tipus boolean que anomena *FirstTime* i que indica si és la primera vegada que es crea l'activitat. És el procediment en el qual s'ha de carregar el fitxer amb el disseny (*.bal*). En el cas de l'exemple anterior, carrega *Layout1.*

Activity_Pause() es crida quan una altra activitat es converteix en principal.

Activity_Resume() es crida immediatament després de que acabi *Activity_Create* o després de reprendre una activitat prèviament pausada.

L'esquema de la Fig. 6.19 pot ajudar en la compressió dels conceptes i mètodes anteriorment descrits.

6.3.4. Interfície – Disseny

Paral·lelament a la descripció de les funcionalitats que s'incorporarien a l'aplicació, va ser necessari pensar en un disseny que s'adaptés a totes aquestes funcionalitats i millorés les característiques que tenia el de l'aplicació inicial. Destinar esforços i temps del projecte a generar una aplicació mòbil pròpia va ser una elecció meditada i amb uns objectius clars: es volia fer més funcional l'aplicació, obtenir una estètica que tingues consonància amb el robot BB8 i millorar aquelles característiques que l'aplicació inicial tenia. A més a més, tenir una aplicació pròpia permetia poder-la adaptar molt més a futures millores i controlar tots els aspectes d'aquesta.

Es volia obtenir una aplicació més funcional ja que durant l'ús de l'aplicació inicial es van tenir dificultats de control. Per exemple, no es podia conduir amb les fletxes i augmentar còmodament la velocitat. Els botons d'interacció quedaven a prop uns dels altres i es clicaven sense voler-ho. Si es pensava en refer l'aplicació, era important adaptar el disseny d'aquesta a una estètica adequada pel projecte. Estava clar que l'aplicació base estava creada per funcionar i incorporar moltes utilitats, però no per tenir un disseny massa treballat. És per això que un objectiu era *"obtenir una estètica que tingues consonància amb el robot BB8*". Pel que fa a les característiques a millorar, es va partir de la idea d'aprofitar moltes coses que l'aplicació inicial tenia: selecció de dispositiu *Bluetooth*, icona d'estat amb color vermell/verd, tipus d'informació a enviar a l'*Arduino* (*F* per *forward*, *B* per *back*, etc.) i entre d'altres. Es considerava que l'aplicació inicial ja feia el que s'esperava, però tenia moltes possibilitats de millora i podia ser un petit projecte dins d'aquest treball.

Primerament, es van fer uns dissenys en paper amb les idees inicials pel moviment i totes aquelles característiques que es volien incorporar. A la Fig. 6.21 es veuen aquests esbossos.

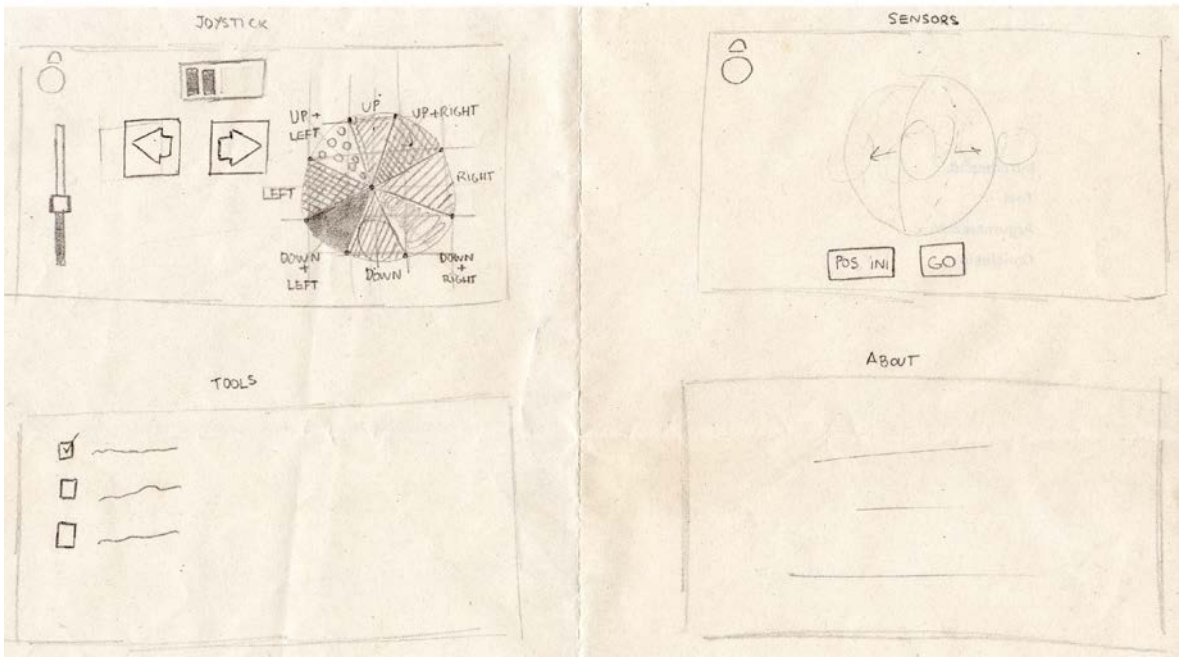

Fig. 6.21. Esbossos en paper de les primeres idees pel disseny. Font pròpia.

Com es pot veure, en un principi es pensava en una aplicació amb quatre pantalles: *joystick*, sensors, eines o *tools* i *about*. Les finestres d'eines i *about* eren les menys treballades: la d'eines s'esperava que fos un espai per seleccionar o habilitar característiques extres del robot com llums, so, càmera, etc; la pantalla d'*about* havia de ser un espai de reconeixement per l'autor, un lloc per escriure el *software* utilitzat i els drets d'autor. Després hi havia la pantalla de sensors, que havia de ser un lloc per poder ordenar al robot que es posés a una posició inicial definida, i també permetre posicionar-lo en l'espai d'una manera concreta. Com se sap, fins a dia d'avui el projecte no incorpora cap d'aquestes funcionalitats, així que aquestes pantalles, en la versió final, estan buides, tot i que apareixen creades en el projecte i s'hi pot accedir.

L'última pantalla és l'anomenada *joystick,* que permetia moure el robot: es van agafar els elements que apareixen en l'aplicació inicial, però reorganitzats en l'espai. S'hi van incorporar noves possibles funcionalitats, com l'estat de les bateries o el gir del cap, i es va repensar la manera de moure el robot. En comptes d'utilitzar fletxes, es va optar per un sistema més còmode i que permetia el control del robot amb un únic dit; un *touchpad* o panell tàctil circular. Físicament no és un panell tàctil com el que incorpora un ordinador portàtil, però funciona d'una manera similar per marcar la direcció que ha de prendre el robot. Per obtenir la direcció que l'usuari vol establir, es divideix el cercle en 8 trossos, i a cada àrea que aquests defineixen se li assigna una direcció. Així doncs, quan l'usuari es mou pel *touchpad,* aquest actua com un *joystick* i s'obté una direcció. Aquest mètode permet a l'usuari poder disposar de l'altre mà per canviar la velocitat, moure el cap, etc.

Un cop es van fer aquests esbossos, es van generar els dissenys preliminars a ordinador. Per complementar els dissenys, es van utilitzar imatges gratuïtes i es van crear components específics. L'aplicació va ser dissenyada en la seva totalitat sense utilitzar dissenys de tercers o components amb drets d'autor. A la Fig. 6.22 veiem un recull d'aquests dissenys preliminars.

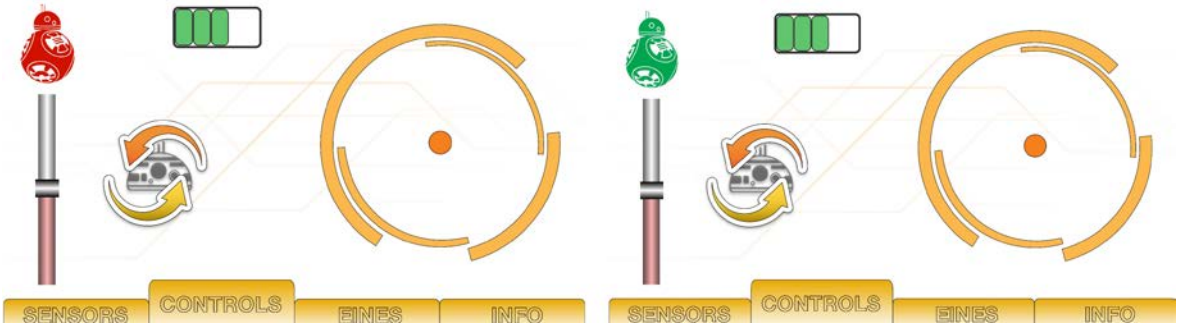

Fig. 6.22. Dissenys preliminars fets amb ordinador. Font pròpia.

Com apareix a la figura anterior, aquests dissenys ja establien la paleta de colors a utilitzar i alguns components, com l'indicador d'estat de la connexió, fet amb un BB8, o el disseny del *touchpad*. Amb aquests dissenys fets, es va utilitzar una plataforma en línia anomenada *marvelapp* [25][26] que serveix per pujar les imatges dels dissenys de l'aplicació i establir relacions entre ells de manera fàcil. Això possibilita visualitzar com aquests dissenys van canviant i evolucionant i, a més a més, permet veure si hi ha errors o si falten components per definir.

Adaptant-se a les funcionalitats reals i als límits establerts del projecte, es va anar desenvolupant la versió definitiva de l'aplicació i, conseqüentment, del disseny. A la Fig. 6.23 es presenta un conjunt d'imatges que mostren l'aspecte actual de l'aplicació.

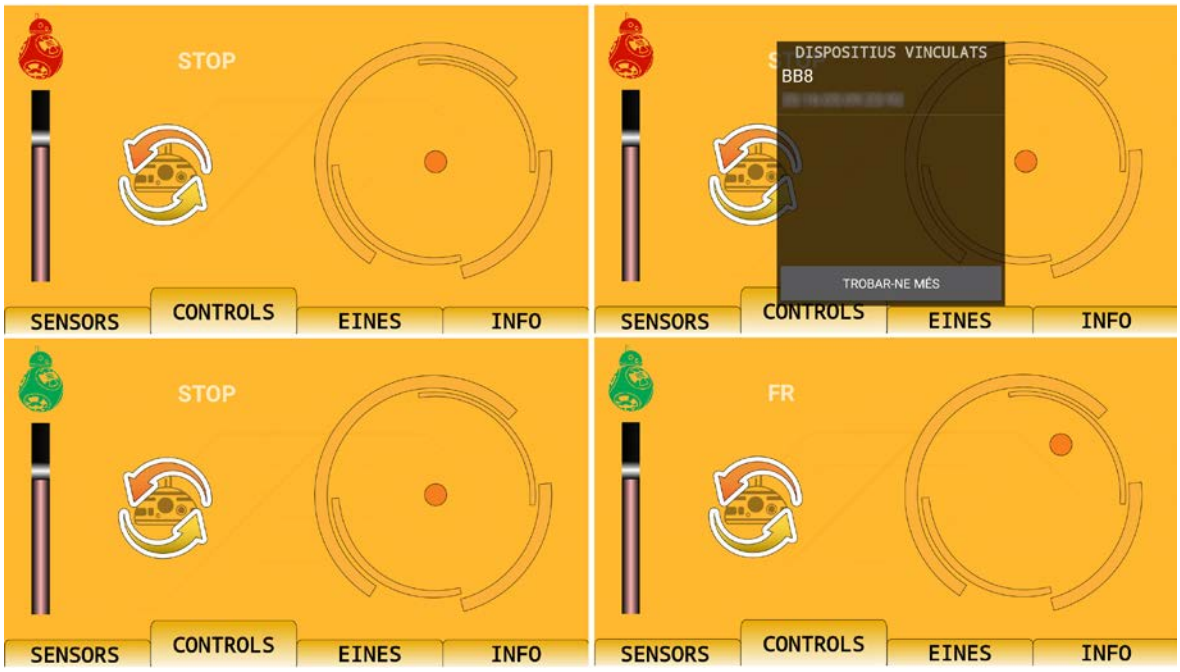

Fig. 6.23. Dissenys actuals de l'aplicació. Font pròpia.

L'aspecte final guarda moltes similituds amb l'aplicació inicial, ja que realment aquesta s'ha utilitzat com a plantilla i guia. En la imatge inferior esquerra, es veu el *touchpad* en funcionament i un text que mostra quina direcció marca aquest (*FR – forward right – endavant a la dreta).* La barra vermella lateral, que es troba sota la icona d'estat vermella/verda, és el regulador de velocitat.

6.3.5. Funcionalitats que incorpora

Abans de començar aquest punt, es vol senyalar que molts conceptes descrits en l'apartat 6.2.2 són igualment útils i aplicables per aquesta explicació. Per evitar reiterar en l'explicació s'evitarà parlar altre cop dels tipus de variables i de les estructures bàsiques. A més a més, la manera en la que s'ha comentat el nostre codi segueix les mateixes directrius que les del codi *Arduino*.

Les variables utilitzades i les funcionalitats generades per tal de fer funcionar l'aplicació estan íntegrament comentades en el codi que s'adjunta a l'annex. Aquest és el codi font real de l'aplicació, però afegint un conjunt de comentaris per tal de facilitar la seva lectura. Amb l'explicació del codi font del programa d'*Arduino* de l'apartat 6.2.2, ja s'ha detectat que tenir un codi escrit i adjuntar les imatges d'aquest mateix pot quedar repetitiu, i més si els comentaris són suficientment explicatius. Així doncs, la intenció d'aquest apartat serà, primerament, explicar el funcionament genèric de l'aplicació, com es va fer amb el programa d'*Arduino*, i, posteriorment, concretar les funcions generades i explicar la interacció de les unes amb les altres.

Un cop l'aplicació s'executa, mostra la pantalla "controls" (*joystick*); aquesta és l'activitat *main* o principal. Tot seguit, internament, es declaren i defineixen totes les variables globals tant de l'aplicació com de l'activitat *main*. Per tot això, només obrint-la ja demana permís per habilitar el *Bluetooth*. Un cop carregada la interfície gràfica i inicialitzades les variables, l'aplicació queda en espera pendent de l'ordre que li doni l'usuari. En aquests moment, la icona d'estat en forma de BB8 està de color vermell; fent clic sobre aquesta es poden buscar els dispositius *Bluetooth* pròxims o connectar-se a un dispositiu ja aparellat; aquests es mostren a través d'una llista desplegable. Si la connexió és satisfactòria, la icona d'estat es torna verda. En aquest moment, qualsevol moviment en el *touchpad* de la dreta o en la barra reguladora de velocitat de l'esquerra s'enviarà per *Bluetooth* al dispositiu del robot. Aquest interpretarà el que li arribi i actuarà en conseqüència. El conjunt de possibles instruccions que s'envien són les ja comentades durant la programació del codi amb *Arduino*, i no deixen de ser les mateixes que les que enviava l'aplicació gratuïta que inicialment s'utilitzava.

Principalment hi ha tres classes ja creades pel propi entorn, que són indispensables, sent gairebé les més importants: *BluetoothAdmin, Serial* i *AsyncStreams*. La combinació d'aquestes classes fa possible establir la connexió amb un dispositiu extern i poder enviar i rebre dades. Aquestes s'analitzaran i comentaran per separat, tot veient quines funcions se'n deriven.

BluetoothAdmin és una classe que permet administrar l'adaptador *Bluetooth* del dispositiu en el que executa l'aplicació. Amb aquesta es pot habilitar o deshabilitar l'adaptador i monitoritzar el seu estat. També permet descobrir nous dispositius que es trobin a l'abast; a través del mètode *StartDiscovery*. Els esdeveniments associats a aquesta classe que s'utilitzen en el codi són:

- *DeviceFound*: S'executa quan, durant el procés de descoberta de dispositius (*StartDiscovery*), es troba un nou dispositiu a l'abast.
- *DiscoveryFinished*: S'executa un cop ha finalitzat el procés de descoberta de dispositius.
- *StateChanged*: S'executa quan l'estat de l'adaptador canvia. Per exemple, quan passa d'activat (*ON*) a desactivat (*OFF*).

Serial és una classe que permet establir connexió amb altres dispositius *Bluetooth* a través d'un port sèrie virtual. Els esdeveniments associats a aquesta classe que s'utilitzen en el codi són:

- *Connected:* S'executa quan, després d'intentar establir connexió a través del mètode *Connect(),* es rep una resposta. La resposta queda registrada a la variable d'entrada *Success* de tipus *bool.*

AsyncStreams és una classe que permet llegir i escriure dades, a través dels components *InputStream* i *OutputStream*, en segon pla; sense bloquejar el fil principal de l'aplicació. Aquesta classe s'ha d'inicialitzar a través del mètode *Initialize* per associar-se amb un component de classe *Serial*. Aquest és el que pot definir un *InputStream* i un *OutputStream*. Feta la inicialització, a través del mètode *Write,* podem enviar una cadena de bytes. Així doncs, per enviar un missatge serà necessari el seu processament previ per obtenir els bytes equivalents. Els esdeveniments associats a aquesta classe que s'utilitzen en el codi són:

- *NewData*: S'executa quan es rep un missatge. El format de rebuda és en bytes.
- *Error:* S'executa quan la classe detecta un error. En el codi, s'utilitza per detectar si s'ha perdut la connexió.

Explicades aquestes funcions, es continuarà l'anàlisi del codi amb la resta de procediments.

Tots els esdeveniments que acaben amb un *_Click* són procediments que s'executen quan es fa clic sobre l'objecte que el defineix. Per exemple, l'esdeveniment *BluetoothIcon_Click* s'executa en fer clic en la imatge (icona) *BluetoothIcon.* En l'aplicació aquesta és una imatge d'un BB8 que indica l'estat de la connexió, depenent del color que té. Aquesta icona es pot veure a la Fig. 6.23.

De manera similar, els esdeveniments que acaben amb un *_Touch* són procediments que s'executen quan es toca o es polsa sobre l'objecte que els defineix. En aquest cas, l'exemple seria *BTOut_Touch. BTOut* és un component tipus *Panel* que ocupa tota la pantalla i és invisible. Aquest agrupa *BTPanel, BTLabel, BTList* i *BTMore* que són elements que només apareixen en el procés de selecció o cerca de nous dispositius *Bluetooth*.

Com veiem a la Fig. 6.24 el panell *BTPanel* (en color taronja) agrupa *BTLabel, BTList* i *BTMore;* qualsevol pulsació fora d'aquesta zona serà detectada en el panell *BTOut* (en color vermell). Així doncs, la tasca real d'aquesta funció és descobrir si es fa un clic a l'exterior.

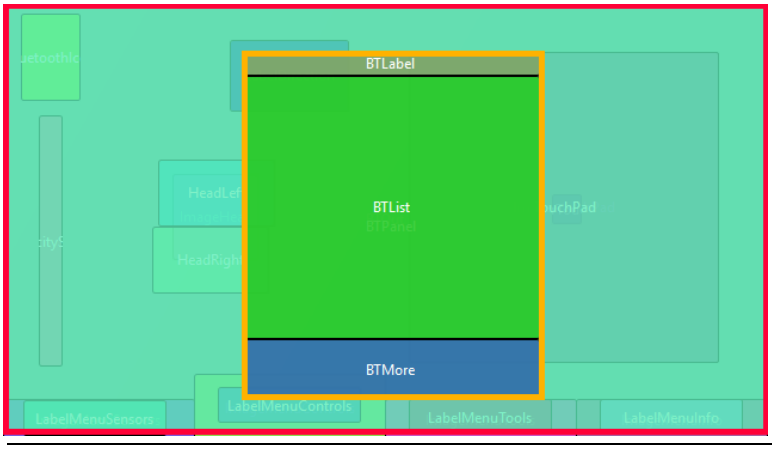

Fig. 6.24. Capes dels components de l'aplicació en mode dissenyador. Font pròpia.

El mòdul *VSeekbar* és un mòdul estàtic de funcions extern que permet generar barres lliscants. Aquest es va extreure del fòrum que té la comunitat de B4A. El mòdul està sotmès a drets d'autor i éss per això que s'ha adjuntat en el punt [27] de la bibliografia. Aquest mòdul no conté comentaris propis, però ja inclou alguns comentaris de l'autor. *VSeekbar* defineix, en l'activitat *main*, un esdeveniment acabat amb *_ValueChanged*. Aquest s'executa quan el valor de la barra lliscant canvia.

Els procediments que resten a comentar són funcionalitats que es criden des d'altres procediments però que no són esdeveniments associats a cap component; són funcions creades per l'autor del codi font.

Els procediments *InitializeGlobalVariable* i *InitialValueGlobalVariables* són funcions que inicialitzen les variables globals de l'activitat i els hi associa un valor inicial. Aquest tipus de funcions es creen per agrupar mètodes similars i deixar més simplificat el procediment que les executa.

Els procediments *ResizeImageView* i *ResizeLabel* serveixen per redimensionar elements de tipus *ImageView* i *Label* que reben agrupats en una llista, a través de l'atribut d'entrada de tipus *List.* Cada element de la llista passa per un bucle per redimensionar-se i posicionar-se de nou.

Queden per comentar 5 procediments: *notInList, SendBluetooth, GesturesTouch, GetDirection* i *GetPairedList.* D'aquest grup, els més simples són *notInList* i *GetPairedList.* El primer comprova si un element ja existeix en la llista *BTList*. Aquesta és la llista on es guarden els dispositius trobats o aparellats amb l'adaptador *Bluetooth*. La funció fa la comprovació mirant si existeix un element amb la mateix adreça MAC. L'adreça MAC és una adreça que identifica una interfície de xarxa de maquinari gairebé de manera única [28]. El segon procediment, *GetPairedList*, obté i afegeix a la llista *BTList* els dispositius ja aparellats.

SendBluetooth és el procediment que s'utilitza, a través del component *AsyncStreams*, per enviar una cadena de bytes al dispositiu *Bluetooth* receptor. El missatge a enviar l'obté a través del seu atribut d'entrada *packet*. Aquest està en format *String,* així que cal fer una conversió a bytes.

Finalment, els procediments *GesturesTouch* i *GetDirection* són dels més importants i innovadors del codi. El primer permet detectar una pulsació mantinguda en el panell *TouchPad* i, a través de càlculs trigonomètrics, enviar un valor a *GetDirection* perquè aquest interpreti quina direcció està marcant aquesta pulsació en el *TouchPad.* La primera funció també permet limitar el moviment del *CursorTouchPad* a un cercle. Com ja s'ha comentat, la segona funció interpreta la informació que li arriba des de *GesturesTouch*, obté una direcció i envia a *SendBluetooth* un missatge amb el caràcter corresponent (F per endavant, R per dreta, I per endavant a ala dreta, etc.). Els diferents caràcters enviats es poden consultar en el codi font comentat.

7. Impacte ambiental

Com bé s'ha descrit en la memòria, aquest projecte s'ha desenvolupat intentant donar una nova vida a components o peces que qualsevol persona pot tenir per casa o en desús. Això s'ha fet per intentar contribuir a un reciclatge real i generar menys residus en el procés constructiu. En són exemples el paper de diari que es va utilitzar durant la creació del cos, l'ús de retalls de fusta utilitzats per fer les plataformes inicials de l'esquelet intern, l'ús de components reciclats per decorar el cap i els imants que es van treure d'altaveus de música que estaven espatllats. S'han intentat comprar les coses indispensables per fer un pressupost el més ajustat possible, tal com veurem en l'apartat 8, i reduir l'impacte ambiental del projecte.

També cal esmentar el material que s'utilitza per les impressions 3D, que s'ha fet servir per fer a mida diverses peces del projecte. Com ja s'ha comentat a l'apartat 4.3, el material que s'utilitza s'anomena àcid polilàctic i és un polímer termoplàstic que prové de fonts renovables. En ser un termoplàstic, si el component ja no fos útil, seria possible reescalfar-lo i donar-li una nova forma (nova funcionalitat). A més a més, aquest es degrada ràpidament en el medi ambient i té una toxicitat baixa [29][33].

En la cara negativa de la moneda trobem tots els components electrònics o elèctrics que s'han utilitzat, i que són complicats de reciclar o reutilitzar un cop la seva vida útil s'acaba. Parlem d'alguns d'ells a continuació:

Les bateries tipus LiPo tenen un reciclatge molt específic i cal descarregar-les del tot i portar-les a una deixalleria o punt verd pel seu correcte tractament. Tanmateix, el procés de reciclatge de les piles de liti és complicat i costós, i sovint acaben en abocadors comuns. Cal dir també que no són bateries on es puguin separar els components i reutilitzar-los, ja que el cost d'aquest procés és molt superior al preu de fabricació. La vida d'aquestes bateries s'estima en cicles de càrrega i descarrega, tenint en compte que se'n farà un bon ús i que es seguiran les indicacions que ja s'han esmentat en l'apartat 5.3.2.1. Igualment, també es pot fer referencia a les piles no recarregables que es van utilitzar en les primeres versions del robot. Tenen una vida útil menor a les bateries LiPo, ja que no són recarregables i el seu reciclatge també passa per portar-les a un punt verd o contenidor específic.

La placa *Arduino* i els seus components, com la *Shield* o el mòdul *Bluetooth,* també són components complicats de reciclar. Se'ls hi poden trobar altres usos mentre funcionin correctament. Un exemple seria la pròpia placa *Arduino* MEGA 2560, que pot re-compilar i carregar tots aquells programes que l'usuari desitgi, canviant totalment, si s'escau, la seva funcionalitat. A més a més, sovint quan un dispositiu electrònic falla o es crema no queda totalment inutilitzat si no que només hi ha algunes parts o components, més o menys vitals,

que no funcionen bé. Així doncs, en teoria és possible substituir aquest component i continuar treballant amb normalitat. La realitat és que aquest procés implica temps per descobrir l'error concret i sovint cal un nivell de coneixement d'electrònica més elevat del que l'usuari té. Aquests dispositius estan fets per fer un *plug and play* i són tan econòmics que ningú pensa en una reparació o una reutilització dels components per separat. Actualment, en una societat que té tanta informació al seu abast, el que s'hauria de fer és utilitzar les *datasheet* proporcionades per intentar reduir al mínim els residus; ja sigui reparant el dispositiu o intentant guardar els components útils per futures reparacions d'altres dispositius. A la xarxa podem trobar projectes que, fins i tot, proposen la creació de plaques *Arduino* bàsiques amb components reciclats [30].

Finalment, a tot això s'han de sumar els kWh d'energia elèctrica que s'han requerit per totes les màquines elèctriques que s'han fet servir en la confecció del robot i per l'ús de l'ordinador per programar l'*Arduino*, crear l'aplicació mòbil i redactar aquesta memòria. Se sap també que, aquestes màquines, en la seva creació i en un futur generaran residus que, igualment, es podran intentar reaprofitar o que acabaran en un abocador.

Com ve se sap, la regla de les tres erres implica reduir, reutilitzar i reciclar. La regla va per aquest ordre, ja que primer s'hauria d'intentar reduir i no consumir reiteradament tant, segon intentar reutilitzar tot allò que ja s'ha adquirit però pot tenir una nova vida i, finalment, arriba un moment que s'ha de llençar, però intentant reciclar per disminuir l'impacte ecològic del residu. Per intentar millorar l'impacte ambiental del projecte cal analitzar, en primera instància, on s'hauria pogut reduir. Per exemple, es pot pensar en generar els propis dispositius amb components electrònics recuperats d'antics aparells o, si cal comprar, intentar fer-ho tot d'una tirada a una empresa i estalviar en transport i emissions de CO_2. Cal dir que, sovint, es pensa que importar components, per exemple de la xina, resultarà més econòmic, sense tenir en compte que la motxilla ecològica [31] que aquests porten és molt superior a la dels mateixos components fabricats més a prop. En segon lloc, es continuaria intentant reutilitzar tots aquells materials als que se'ls hi pogués donar una segona vida (per decorar el robot, per fer l'interruptor, etc.). Finalment, caldria assegurar-se que se sap com reciclar tots els components del projecte i les possibles afectacions que aquests tindran sobre el medi ambient, per poder prendre així les mesures corresponents quan una peça s'hagi de canviar o es desmantelli totalment el projecte.

8. Pressupostos

Per fer els pressupostos del projecte, primerament s'ha plantejat quantes hores s'ha dedicat a cada part d'aquest i s'ha fet un taula detallada dels preus i quantitats de tots els materials o components comprats per construir el robot. Cal afegir que el total d'hores teòriques és 360, un número que sorgeix de la multiplicació dels crèdits del Treball de Fi de Grau, que són 12, per les hores de dedicació a les que correspon cada crèdit, que són 30.

Primerament, a la Taula 8.1, es presenta el desglossament d'hores invertides en cada part del projecte, que són: la construcció del robot, la creació del programa per l'*Arduino* i la programació de l'aplicació per mòbil.

PART DEL PROJECTE	HORES
Robot	255
Arduino	40
App	65
	360

Taula 8.1. Desglossament de les hores dedicades a cada part del projecte. Font pròpia.

Com es pot veure, la dedicació menor se l'emporta el programa d'*Arduino,* ja que, com s'ha comentat en l'apartat 6.2, el codi font té un tros extret d'un exemple creat per l'empresa que subministra la *shield* adquirida, i un tros reutilitzat del codi del manual que s'ha seguit. La segueix la creació de l'aplicació per mòbil, que sí que va comportar el fet de generar una aplicació de zero i aprendre a fer anar el programa Basic 4 Android. Finalment, trobem la construcció pròpiament del robot, que té una dedicació aproximada de 255 hores respecte les 360 totals, el que suposa un aproximadament el 70% del temps. Això és degut a que tot s'ha creat a mà i, sovint, algunes parts s'han hagut de repensar o fer de nou. Seria el cas del cap, que es va haver de refer, o la multitud d'idees per fer la finestra.

A continuació, a la Taula 8.2, es presenta un desglossament de la pròpia construcció del robot, dividida en les parts que s'han presentat en aquesta memòria: part mecànica, part elèctrica i part electrònica.

PART	HORES
Mecànica	160
Elèctrica	25
Electrònica	40
	225

Taula 8.2. Desglossament de les hores dedicades a cada part de la construcció del robot. Font pròpia.

S'ha considerat que la part elèctrica i electrònica havien de tenir un pes menys important, ja que tota la part del disseny i distribució dels components s'anava adaptant a través dels canvis de la part mecànica. Així doncs, aquests només s'havien de col·locar i fer una correcte instal·lació; segura, ordenada i optimitzada. Igualment també s'ha de comptar la part de trenar el cablejat, soldar alguns components o fer les proves corresponents amb cada component. Per tot això, a la part mecànica se li atribueix un pes del 63%, que correspon a les 160 hores de construcció real que s'han establert.

Es presenta doncs a la Taula 8.3 el pressupost total del projecte.

CONCEPTE	QUANTITAT	UA	PREU UNITARI	UA	COST FINAL
Mà d'obra d'enginyer	360	hores	40	€/h	14.400,00 €
Consultes agents externs	50	hores	40	€/h	2.000,00 €
Materials utilitzats					238,37 €
Transport materials	25	% material	238,37	€	59,59 €
Llicencies programes					283,00 €
Despeses extraordinàries					1698,10 €
					18.679,06 €

Taula 8.3. Pressupost total del projecte. Font pròpia.

A continuació es detalla a què correspon cada partida establerta en el pressupost:

- Mà d'obra d'enginyer: Són els diners que ha de cobrar l'enginyer per la dedicació d'hores que s'ha establert. Els seus honoraris s'ha establert a 40 €/hora.
- Consultes agents externs: Són els diners reservats per pagar les consultes realitzades a persones externes al projecte, per fer estudis o per dissenyar algunes parts. Alguns exemples serien l'ajuda rebuda durant la confecció de les peces 3D fetes a mida o les consultes realitzades per intentar solucionar els problemes que es van detectar al tancar l'esfera. També s'ha establert un preu de 40€/hora.
- Materials utilitzats: És el cost total de tots els materials que s'han utilitzat en el projecte. En la Taula 8.4 es presenta el desglossament d'aquest partida.
- Transport materials: És un valor que s'ha establert a causa d'haver adquirit la major part de les peces en comerços en línia i a diferents distribuïdors. S'ha considerat una bona aproximació valorar aquesta en el 25% del cost dels materials.
- Llicencies programes: Pel desenvolupament del projecte s'han utilitzat els següents programes:

- SolidWorks pel disseny 3D. Un any de llicència costa gairebé tant com tot el projecte. Donat que utilitzar-lo és opcional i que es va utilitzar principalment per la modelització de les impressions 3D que van ser un afegit posterior; ho comptabilitzarem com una llicència d'estudiant a 90 €.

- Basic 4 Android per la creació de la aplicació mòbil. Una llicència vitalícia costa 50 €.

- Microsoft Office Word – Excel per la confecció de la memòria i documentació. Un any de llicència costa 59 €.

- Autodesk Eagle pel disseny i documentació gràfica dels circuits elèctrics. Es va fer ús d'aquest programa ja que tenia una gran quantitat de llibreries disponibles amb components esquematitzats. Un any de llicència costa 84 €.

- Despeses extraordinàries: És la partida destinada a assumir les possibles desviacions respecte el pressupost inicial. Està comptabilitzada com un 10% del total obtingut per la suma de les altres partides.

PART	IDENTIFICADOR	DESCRIPCIÓ	UNITATS	PREU UNITARI	PREU TOTAL
Mecànica	TAMIYA-70111	Rodes de goma amb llanta blanca de D 56mm i W 25mm	1	7,90 €	7,90 €
Mecànica	POLOLU-1084	Suports d'alumini en forma de L 37,6x49,2x37,6mm	1	6,01 €	6,01 €
Mecànica	Arduino Robot PCB Spacer	5x Separador hexagonal M3 10mm mascle-femella	2	1,50 €	3,00 €
Mecànica	Roll-on	Desodorant tipus roll-on	9	0,79 €	7,11 €
Mecànica	Esfera porexpan	Esfera porexpan 200mm	1	7,95 €	7,95 €
Mecànica	Aguaplast	Aguaplast fibra	1	3,60 €	3,60 €

Mecànica	Aguaplast	Aguaplast fusta	2	3,85 €	7,70 €
Mecànica	Cola Blanca	Cola Blanca 1 L	2	5,43 €	10,85 €
Mecànica		Pilota inflable platja 360mm	1	2,50 €	2,50 €
Mecànica		Tela canvas (en 1,5 x ? m2)	2	9,00 €	18,00 €
Mecànica	TITANLUX Color	Pintures TITANLUX Blanc,Taronja,Negre	3	3,85 €	11,55 €
Mecànica		Cable 2 fils (en m)	2	1,00 €	2,00 €
Mecànica		Regleta blanca 2 borns	6	0,08 €	0,48 €
Mecànica	2 CTS R.5MM	Regleta taronja de 2 pins a 2 borns	3	0,54 €	1,62 €
Mecànica	Hettich magnet	Imants neodimini D 30mm W 7mm	6	1,30 €	7,80 €
Mecànica		Cinta americana plata 10m	1	2,95 €	2,95 €
Mecànica		Cinta plàstic vermella	1	1,00 €	1,00 €
Mecànica	Pin header PCB	5x Tira 40 pins mascle per soldar	1	1,14 €	1,14 €
Mecànica		Spray blanc	3	5,85 €	17,55 €
Mecànica		Paper de lija	2	0,45 €	0,90 €
Electrònica	VNH2SP30	Driver Module 30A Motor Shield L298N	1	16,50 €	16,50 €
Electrònica	HC-05	Bluetooth Màster/Esclau Arduino	1	4,90 €	4,90 €

Electrònica	Cable	Cable USB tipus A/B	1	2,73 €	2,73 €
Electrònica	Arduino MEGA	Arduino MEGA 2560 R3	1	14,95 €	14,95 €
Elèctrica	POLOLU-1102	Motor 37D 19:1 12VCC 500rpm	2	21,77 €	43,54 €
Elèctrica	FUTABA S3003	Servomotor 360 graus gir continu	1	7,99 €	7,99 €
Elèctrica	R6 LR06 4+4	Portapiles 8x AA 12V	1	1,99 €	1,99 €
Elèctrica	LiPo Bateries	Bateries LiPo 7,4V 2500 mAh	2	10,33 €	20,66 €
Elèctrica	LiPo Charger	Cargador per bateries LiPo 7,4V 2500 mAh	1	3,50 €	3,50 €
		TOTAL			238,37 €

Taula 8.4. Taula del pressupost del material utilitzat. Font pròpia.

Un cop s'ha presentat el pressupost, cal fer un anàlisi i reflexió sobre aquest. El procés de construcció del robot té un cost aproximat de 238,37€, que s'acosta bastant a l'objectiu inicial d'obtenir una memòria que detalles els passos per poder fer un robot més o menys econòmic, comptant només amb els coneixements personals. Igualment, quan es mira tot el projecte i s'ha de fer una valoració econòmica més realista comptabilitzant les hores invertides i tots els recursos invertits en aquest, es pot veure que el projecte s'eleva a gairebé 19.000 €.

En realitzar el pressupost general, s'ha detectat que hi ha partides que es podrien disminuir per tal d'intentar abaratir tot el projecte. Podria ser un exemple fer ús de *software* lliure, substituint així tots els programes de pagament. Possibles substitucions serien: SolidWorks per Blender, Autodesk Eagle per Kicad, Microsoft Office per OpenOffice, etc.

També es podria intentar ajustar molt més el cost del material i el transport d'aquest, intentant comprar tots els components de cop i a un únic subministrador. Aquest fet permetria reduir molt el cost de transport i intentar obtenir un preu pactat pel conjunt.

Tanmateix, les parts importants del pressupost, com són el sou d'un enginyer o el de les consultes externes, no es pot intentar retallar, ja que per molt que carreguin amb bona part del pressupost, són necessàries unes bones condicions laborals i salarials si es vol obtenir un bon resultat final. Es podria mirar d'arribar a un acord o valorar si surt més a compte no treballar per hores si no pactar un preu total del projecte, però llavors cal establir bé quina serà la magnitud del projecte (abast) i detallar bé les condicions d'entrega d'aquest.

Finalment, per tancar l'apartat de pressupost, es vol comentar que pot semblar estrany no incloure en els materials el cost de les impressions 3D. S'ha optat per no comptabilitzar-la, ja que augmenta bastant el cost del projecte de construcció i realment, en molts casos, es pot prescindir d'aquesta tecnologia. Els eixos, per exemple, s'haurien pogut estalviar comprant rodes amb adaptadors de l'eix correctes. Tot l'esquelet intern va ser una optimització que es va fer un cop es va tenir el model de fusta, amb l'objectiu d'optimitzar el màxim el robot; sense aquesta millora el robot ja era funcional. La finestra i el marc si que són components més complicats de fer manualment, però tampoc necessaris pel funcionament del robot. Estan fets per garantir una obertura ràpida, però el robot sempre s'haurà d'obrir per la meitat per reparacions majors.

En qualsevol cas, si es volgués tenir en compte el cost d'aquest, s'hauria de calcular el volum dels models 3D, multiplicar-los per un factor corrector de 0,8 – 0,75, ja que a l'imprimir-los la impressora optimitza el material i, a través de la densitat del material plàstic[3] i del preu per gram[4], s'obtindria el cost total.

[3] Densitat de 2,75 g/cm^3 , en aquest cas.

[4] Preu aproximat entre 0,05 – 0,10 €/g.

9. Resultats, dificultats, tests i validació

En ser un projecte que tenia una part molt important pràctica i manual, s'han hagut d'anar solucionant tots els problemes que han anat sorgint a mesura que es detectaven.

Un dels primers problemes va ser l'adquisició dels materials elèctrics i electrònics que el manual que es va fer servir de guia proposava. Es va intentar cercar components similars perquè se sabia que, amb aquests, el projecte podria funcionar. Al final és com una recepta on hi ha les quantitats i els ingredients, és obvi que aquesta recepta ha estat modificada i que s'ha afegit tot allò que es creia convenient i podia millorar el projecte i, fins i tot, s'ha hagut d'adaptar coses i acabar-la de completar. Tanmateix, els fonaments eren necessaris per no anar amb inseguretat fins al final. És per això que es van intentar adquirir els mateixos motors, de la marca POLOLU, i es van utilitzar distribucions i estructures similars pels components interns. El fet que la guia estigui feta per una persona de Filipines fa que alguns components no siguin tan fàcils d'aconseguir o siguin més cars. Per exemple, els motors es van poder aconseguir a un bon preu, però la *shield* que proposava la guia es va haver de substituir per una més econòmica amb funcionalitats similars, ja que el cost era massa elevat aquí. Igualment, també feia falta comptar amb els terminis d'enviament que tenien els components. Alguns podien arribar amb una o dues setmanes, però sovint, si es compren dispositius a Xina per estalviar en el preu, triguen al voltant de un mes, i feia falta anar amb molt de compte per intentar fer una comanda i no descobrir després que es necessitava un nou component.

Hi ha hagut un problema de toleràncies i precisió pel fet de fer alguns components manualment. El cos, per exemple, és esfèric però no es pot esperar que sigui una esfera perfecta, ja que la pilota es va anar desinflant i les arrugues i altres detalls van generar imperfeccions. Això va fer que,al principi, un cop es van tallar les dues meitats d'esfera, es veiés complicada la seva unió posterior al afegir pes a la base. Al final, la unió ha estat bastant exacte i està ben sallada amb la cinta americana. El problema real d'aquesta imperfecció inevitable ha estat a l'hora de generar l'esquelet interior. Quan es va realitzar el primer esquelet en fusta, van haver-hi molts problemes amb les columnes per fer quadrar les dues meitats i, posteriorment, quan es va fer tot amb impressió 3D és va voler ajustar al màxim les dimensions però, com era d'esperar, l'estructura necessitava petits retocs. Com s'ha explicat en l'apartat 4.6, es va optar per substituir les columnes de plàstic per les antigues de fusta, que es podien tallar millor, i afegir-hi escuma en els extrems per poder adaptar més exactament l'altura requerida pel correcta tancament. També, com ja s'ha dit en diversos punts de la memòria, el cap del robot es va haver de repetir, ja que les dimensions no eren les correctes i no es va detectar fins després de pintar-lo tot.

La finestra també és un clar exemple d'assaig i error d'aquest projecte. S'ha modelat amb tot tipus de tancaments i s'ha intentat dimensionar-los per complir amb la seva utilitat però ser igualment robustos. Tanmateix, un cop implementada la solució final, es va valorar que tenia límits i es va concloure que era necessari que el robot conserves el seu accés a través de la separació de les dues meitats; per això el tancament es va fer finalment amb cinta americana.

Un cop tancat, a més a més es va veure un problema de fricció de les rodes. Aquest problema es va detectar tard, ja que tot i fer proves anteriorment i que funcionessin correctament, aquestes es van realitzar amb l'esquelet de fusta. La decisió de canviar l'esquelet de fusta per un imprès es va prendre per optimitzar la distribució dels components i obtenir un esquelet millor. Tanmateix, durant aquest canvi, no es va tenir en compte la gran disminució de pes i que això podia afectar al moviment del robot. Es van buscar solucions per intentar augmentar la fricció de les rodes, tant a través de afegir pes a la plataforma com de canviar el material de contacte de les rodes amb l'esfera interior.

Més enllà dels problemes mecànics, també van sorgir problemes en el camp electrònic i elèctric. Electrònicament, es va demanar ajuda al tutor del treball per intentar solucionar un problema amb els impulsos PWM que l'*Arduino* enviava a la *Shield*. Tal com surt detallat al codi que s'explica a l'apartat 6.2, es va haver de modificar manualment la freqüència dels pins corresponents a través de forçar un valor concret d'un registre del *Arduino*. La idea per fer aquest canvi va ser suggerida a través d'un fòrum on es discutia aquest mateix problema, i es va buscar algun recurs en línia que expliqués com fer el canvi [14][32]. En la part elèctrica hi van haver algunes dificultats per calcular amb exactitud els valors d'amperatge que requeria el circuit per funcionar correctament. Tot i fer servir les eines del laboratori, era necessari saber els valors quan el robot estava en moviment i això dificultava prendre aquesta mesura.

Finalment, en la part de programació es van fer diverses proves. Sovint quan es programa va bé desenvolupar petites funcionalitats, anar-les provant i augmentar el que fan quan no tenen errors. Amb l'*Arduino,* el codi en general estava clar i, menys algun detall que va requerir més atenció, com és el cas del *pull up* (apartat 6.1.1.3), la resta es anar desenvolupant sense massa dificultat. Sempre provant primer el component per separat i després la interacció amb la resta. Amb el Basic 4 Android hi va haver un petit procés d'aprenentatge, un de disseny de l'aplicació i finalment la implementació d'aquesta. La dificultat estava en trobar bons exemples que servissin de guia per desenvolupar aquelles funcionalitats que s'havien pensat, però com en tot, va ser anar provant.

Com ja s'ha comentat, es van dur a terme tests a cada petit pas del projecte. Sempre que hi havia la possibilitat es van intentar validar els sistemes de control i els funcionalitats a les que havia de respondre el robot. Un exemple seria la generació d'un petit programa específic,

realitzat per provar si el cap girava correctament sobre l'esfera, tot i que aquesta encara no estigues tancada. Les proves van ser un èxit però més endavant es va detectar que, degut al pes del conjunt de l'estructura tancada, seria necessari, en un futur, dedicar-hi més esforços per acabar de fer que funciones correctament i implementar, en l'aplicació mòbil i en les funcionalitats de l'*Arduino*, la possibilitat de moure el cap. Igualment, també es va fer proves de teledirecció de tot l'esquelet intern del robot pel terra pla per comprovar el seu correcte funcionament i control amb l'aplicació desenvolupada.

Finalment, la validació del comportament del robot es va realitzar un cop tancat i solucionats els problemes detectats. Es va teledirigir el robot en un recinte controlat i pla per veure si responia als estímuls que rebia. Es va comprovar que la finestra suportés les forces de contacte amb les rodes i que el cap estigues en el seu lloc en tot moment. També es va intentar fer un ús intensiu del robot per descarregar-lo totalment i tornar-lo a carregar, i així poder comprovar el correcte funcionament de les bateries. Tot i això, com ja s'ha comentat en aquest apartat, hi van haver algunes dificultats després de tancar el robot i algunes d'aquestes validacions no s'han superat tan satisfactòriament com en un principi s'esperava.

10. Futures millores

A l'inici del projecte, es va haver de definir quin era l'abast d'aquest. Això permetia limitar el que s'havia de realitzar per donar per tancat el treball i, a la vegada, separava aquelles coses que estarien implementades i aquelles que quedaven fora. El conjunt de coses que ja se sabia que quedarien fora, juntament amb totes aquelles idees que durant el desenvolupament del propi projecte s'han anat pensant, queden recollides en aquest apartat. Els pròxims avanços passarien per saber-les prioritzar i acabar valorant si la implementació és viable. S'ha realitzat una llista amb les idees, agrupades en les tres parts en les que està separada la memòria: part mecànica, part electrònica i part elèctrica.

Independentment, abans de realitzar qualsevol millora o nova implementació en el robot, és necessari repassar les dificultats que aquest presenta a hores d'ara en el moviment i trobar-hi solucions. Un cop solucionats els problemes es podrà validar el correcte funcionament i ja hi haurà via lliure per, ara si, aplicar totes les millores presentades a continuació.

10.1. Millores mecàniques

Millorar el tancament del cos del robot

- Millorar el acabat superficial final.
    - Repintar la zona de tall i la finestra.
    - Realitzar una última capa protectora amb cola.
- Imprimir tota l'estructura en 3D amb un tancament fet a mida.

Millorar la interacció del cap amb els imants

- Revisar el pes de les dues parts i valorar l'ús d'un servomotor més potent.

10.2. Millores elèctriques

Fer un sistema mesurador de la càrrega restant de les bateries.

Valorar l'ús d'un altre tipus de bateries més sostenibles.

- Ús d'energies renovables o que generin menys residus.

- Generació d'electricitat a través d'altres mecanismes.
- Utilitzar fonts renovables per generar l'electricitat emmagatzemada a les bateries.

Dissenyar un sistema de càrrega sense fils.

10.3. Millores electròniques

Possibilitat d'actualitzar el programari de l'*Arduino* sense connexió directa.

Posicionament del robot (a traves de reflectors interiors).

- Detectar la posició inicial.

Implementació de sensors o altres petits components al cap.

- Emetre sons.
- Afegir LEDs.
- Sensors de proximitat.
- Web cam.

Millorar l'aplicació per mòbil.

- Generar una distribució per iOS i per altres sistemes operatius.
- Optimitzar l'aplicació per altres dimensions de pantalla.

Adaptació del software d'*Arduino*.

- Anar-lo millorant a mesura que s'afegeixen noves funcionalitats o components.

Millorar la comunicació amb l'exterior.

- Explorar les possibilitats de transmissió de dades del *Bluetooth.*
- Implementar un mòdul WIFI per enviar dades o imatges a un dispositiu.
- Reconeixement de veu i control amb aquesta.

Conclusions

Aquest treball tenia tres objectius clars. El primer era dissenyar i construir un robot BB8 funcional: aquest objectiu ha estat assolit per una part, ja que partint de zero s'ha obtingut un artefacte dotat de moviment, controlat remotament i amb una estètica molt semblant a la del model original. Tanmateix, cal recordar que li queden algunes correccions per funcionar correctament i obtenir un robot realment acabat. El segon objectiu era realitzar un informe detallat de tots els passos seguits: aquest objectiu s'ha assolit a través de la realització d'aquesta memòria, juntament amb els annexos. La memòria està escrita per poder esdevenir una guia completa que permeti realitzar el robot des de zero. Aquesta inclou imatges de tots els processos i permet a qualsevol persona aprendre dels errors comesos i prevenir-los. Finalment, el tercer objectiu era obtenir un pressupost ajustat per fer possible la reproducció del projecte per qualsevol interessat/da: aquest objectiu va molt lligat amb l'anterior, ja que descriu que una finalitat important del projecte ha estat realitzar un desenvolupament viable per qualsevol persona. Considerant un cost dels materials de 238,37 €, tal com es detalla a l'apartat Pressupostos, es pot considerar que l'objectiu ha estat assolit. Òbviament, si s'analitza el cost total del projecte la xifra no compleix aquests requeriments però, com ja s'ha explicat, es vol analitzar el cost únicament de construcció del robot.

Durant el projecte es recomana fer un bon anàlisi dels riscos i possibles errors, sobretot en projectes fets a mà i sense càlculs exactes, ja que ens pot estalviar cometre errors. És el cas, per exemple, del canvi de l'esquelet de fusta al de plàstic. Aquest canvi s'hauria d'haver validat i testejat abans de escollir-lo com a definitiu.

Igualment, tot i que el projecte es considera tancat i acabat, està clar que es recomana apostar per assolir completament el primer objectiu i obtenir un robot totalment funcional.

Agraïments

Agraeixo l'ajuda que m'ha donat la meva parella i la seva família durant tot el projecte. Ella ha estat la meva fotògrafa, la meva artista i la meva paciència en molts moments. Igualment, agraeixo els consells i l'ajuda a tota la meva família: les idees dels meus pares o l'ajuda manual dels meus avis.

No em puc deixar d'un amic i company de grau, en Victor. Ha estat en tot moment donant-me consells, suport i ajudant-me amb els models i les impressions 3D.

Segurament, l'agraïment més curiós és a una casa que està a Begur. Una casa que m'ha permès està focalitzat en el robot i on tenia totes les eines necessàries per anar-lo desenvolupant.

Finalment estic molt agraït i content amb el meu tutor del treball de fi de grau, en Manuel Moreno. Una persona que m'ha ajudat sempre que ho he necessitat, m'ha animat en tot moment i m'ha permès realitzar un projecte a la meva mida.

Bibliografia

Referències bibliogràfiques

[1] INSTRUCTABLES. Diy life-size phone controlled BB8 droid [http://www.instructables.com/id/DIY-Life-Size-Phone-Controlled-BB8-Droid, 08 de març del 2017].

[2] PANAMAHITEK. ¿Qué es y cómo funciona un servomotor? [http://panamahitek.com/que-es-y-como-funciona-un-servomotor/, 19 d'abril del 2017]

[3] WIKIPEDIA. Lithium polymer battery. [https://en.wikipedia.org/wiki/Lithium_polymer_battery, 18 d'agost del 2017]

[4] GITBOOKS. ERLEROBOTICS. LiPo Bateries [https://erlerobotics.gitbooks.io/erle-robotics-erle-copter/es/safety/lipo.html, 18 d'agost del 2017]

[5] WIKIPEDIA. Datasheet [https://en.wikipedia.org/wiki/Datasheet, 18 d'agost del 2017]

[6] ARDUINO. SOFTWARE. *Arduino* IDE [https://www.arduino.cc/en/main/software, 19 d'abril del 2017]

[7] ARDUINO. REFERENCE. Serial communication pins [https://www.arduino.cc/en/reference/serial, 19 d'abril del 2017]

[8] ELECFREAKS. HC-03/05 Embedded Bluetooth Serial Communication Module AT command set [http://elecfreaks.com/store/download/datasheet/Bluetooth/HC-0305%20serail%20module%20AT%20commamd%20set%20201104%20revised.pdf, 13 de setembre del 2017]

[9] AUTODESK EAGLE. Owerview [https://www.autodesk.com/products/eagle/overview, 13 de setembre del 2017]

[10] ARDUINO. Shields [https://www.arduino.cc/en/Main/ArduinoShields, 8 d'agost del 2017]

[11] ARDUINO. Reference Constants [https://www.arduino.cc/en/Reference/Constants, 13 de setembre del 2017]

[12] CPLUSPLUS. Preprocessor directives [http://www.cplusplus.com/doc/tutorial/preprocessor/, 13 de setembre del 2017]

[13] ARDUINO. Reference [https://www.arduino.cc/en/Reference/HomePage, 13 de setembre del 2017]

[14] ARDUINO INFO. Arduino PWM Frequency [https://arduino-info.wikispaces.com/Arduino-PWM-Frequency, 13 de setembre del 2017]

[15] CPLUSPLUS. Boolean Operations [http://www.cplusplus.com/doc/boolean/, 13 de setembre del 2017]

[16] WIKIPEDIA. Serial Port [https://en.wikipedia.org/wiki/Serial_port, 13 de setembre del 2017]

[17] GRAN ENCICLOPÈDIA CATALANA. Sistema operatiu [http://www.enciclopedia.cat/EC-GEC-0142997.xml, 13 de setembre del 2017]

[18] KIVY. Kivy [https://kivy.org, 13 de setembre del 2017]

[19] PYTHON. Python [https://www.python.org/, 13 de setembre del 2017]

[20] ANYWHERE SOFTWARE. Store B4A [https://www.b4x.com/store.html, 13 de setembre del 2017]

[21] TUTORIALS POINT. Android – Activities [https://www.tutorialspoint.com/android/android_acitivities.htm, 09 de setembre del 2017]

[22] ANYWHERE SOFTWARE. Android Process and activities life cycle [https://www.b4x.com/android/forum/threads/android-process-and-activities-life-cycle.6487/, 13 de setembre del 2017]

[23] ANYWHERE SOFTWARE. Service Modules [https://www.b4x.com/android/forum/threads/service-modules.7542/, 13 de setembre del 2017]

[24] MICROSOFT. LANGUAGE REFERENCE. Visual Basic Language Reference [https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/, 09 de setembre del 2017]

[25] MARVEL. Marvel App [https://marvelapp.com/, 13 de setembre del 2017]

[26] MARVEL. MarvelApp BB8 Control App [https://marvelapp.com/2331f47/screen/24691217, 13 de setembre del 2017]

[27] ANYWHERE SOFTWARE. Vertical, Horizontal and Circle Seekbars [https://www.b4x.com/android/forum/threads/class-vertical-horizontal-and-circle-seekbars.19444/, 10 de setembre del 2017]

[28] HIPERTEXTUAL. Qué es la MAC Address de un ordenador y para qué sirve [https://hipertextual.com/archivo/2014/09/que-es-mac-address/, 13 de setembre del 2017]

[29] WIKIPEDIA. Àcid polilàctic [https://ca.wikipedia.org/wiki/%C3%80cid_polil%C3%A0ctic, 01 de setembre del 2017]

[30] ARDUINO. Arduino Single-Sided Serial Board [https://www.arduino.cc/en/Main/ArduinoBoardSerialSingleSided3, 13 de setembre del 2017]

[31] ECOINTELIGENCIA. ¿Conoces lo qué es la mochila ecológica? [https://www.ecointeligencia.com/2013/02/mochila-ecologica/, 13 de setembre del 2017]

[32] ARDUINO. LEARNING. PLAYGROUND. Adjusting PWM Frequencies [https://playground.arduino.cc/Main/TimerPWMCheatsheet, 12 de setembre del 2017]

Bibliografia complementària

[33] UNIVERSITAT POLITÈCNICA DE CATALUNYA. UPCOMMONS. Estudi de la biodegradació de l'àcid polilàctic en medi normalitzat i en sòl [http://upcommons.upc.edu/handle/2099.1/22598, 01 de setembre del 2017]

[34] SOLIDWORKS. How to buy [http://www.solidworks.com/sw/buy-software.htm, 04 de setembre del 2017]

[35] AUTODESK EAGLE. Subscribe [https://www.autodesk.com/products/eagle/subscribe, 04 de setembre del 2017]

[36] MICROSOFT OFFICE. Buy and Compare Microsoft Office [https://products.office.com/en-us/compare-all-microsoft-office-products?tab=1, 04 de setembre del 2017]

[37] ANYWHERE SOFTWARE. Basic 4 Android [https://www.b4x.com/store.html, 04 de setembre del 2017]